НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ ЗАХИСТУ ІНФОРМАЦІЇ

«До захисту допущено»
В.о. завідувача кафедрою
_____ М.М.Савчук
(підпис) (ініціали, прізвище)
“___”_____20 _ р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки : 113 «Прикладна математика»
(код і назва)

на тему: Порівняльний аналіз сучасних режимів автентифікованого шифрування
_____
_____
_____

Виконав (-ла): студент (-ка) __4__ курсу, групи _ФІ-62_
(шифр групи)

_____Бунін Олександр Андрійович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник Фесенко Андрій В’ячеславович к.ф-м.н._____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент Южакова Ганна Олексіївна к.ф-м.н _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»
Фізико-технічний інститут

Кафедра математичних методів захисту інформації

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки - 113 «Прикладна математика»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедрою

М.М.Савчук
_____
(підпис) (ініціали, прізвище)

«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту

_____Буніна Олександра Андрійовича _____
(прізвище, ім'я, по батькові)

1. Тема роботи Порівняльний аналіз сучасних режимів автентифікованого шифрування_____
_____ ,
керівник роботи ФесенкоАндрій В'ячеславович, к.ф-м.н. _____ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від _____2020____ р. № _____

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи _____
_____
_____
_____

4. Зміст роботи ____аналіз швидкодії на 64 бітній архітектурі різних комбінації схем GCM, EAX, CCM з алгоритмом шифрування AES _____
_____
_____
_____

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) _____

_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Визначення напрямку дослідження	1.09 — 1.10	
2	Опрацювання матеріалу	1.10 — 1.02	
3	Узгодження теми дослідженя	1.02 — 1.03	
4	Написання коду	1.03 — 1.04	
5	Аналіз отриманих результатів	1.04 — 23.05	
6	Формулювання результатів дослідження	23.05 — 26.06	
7	Оформлення роботи	26.05 — 04.06	

Студент _____ _____Бунін О.А._____
(підпис) (ініціали, прізвище)

Керівник роботи _____ ____Фесенко А.В._____
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Кваліфікаційна робота містить: 66 стор., 17 рисунки, 0 таблиць, 18 джерел.

Метою даної роботи є дослідження питань повязанних зі схемами автентифікованого шифрування на основі суматорів. Алгоритми автентифікованого шифрування, які надають нам можливість безпечно взаємодіяти в комп'ютерних системах через конфіденційність і попередження підробок повідомлень, тобто через забезпечення цілосності нашої інформації. Об'єктом дослідження є інформаційні процеси в системах криптографічного захисту. Предметом - властивості комбінованих режимів автентифікованого шифрування на основі суматорів. AEAD, ЗАГАЛЬНА СХЕМА, СУМАТОРИ, АВТЕНТИФІКОВАНЕ ШИФРУВАННЯ, GCM, EAX, OCB, MAC, GHASH

The purpose of this work is to study issues related to the schemes of authenticated encryption based on adders. Authenticated encryption algorithms that enable us to interact securely in computer systems through confidentiality and the prevention of message forgery through ensuring the integrity of our information. The object of research is information processes in cryptographic protection systems. The subject - the properties of the combined modes of authenticated encryption based on combiners.

ЗМІСТ

ВСТУП

Актуальність дослідження.

Практика безпеки комп'ютерних систем і комунікацій давно визнала таку криптографічну проблему [14]: передача повідомлення таким чином, що частина його захищена секретністю, частина залишається незмінною, і все в цілому автентифіковано. У роботі [14] формалізується та досліджується поняття автентифікованого шифрування з асоційовними даними ($AEAD$). Раніше розробники протоколів зверталися до $AEAD$, використовуючи узагальнену схему (як її вперше назвали та дослідили [3]), де один з методів просто склеює схему шифрування (конфіденційность) та код автентифікації повідомлення (MAC). Наприклад, можна зашифрувати текст M, додати заголовок H, а потім отримати MAC з результату. Такі рішення є настільки природними та очевидними, що, здається, уникнуло, що хтось вирішує криптографічну проблему самостійно. Що спричинило визнання $AEAD$ як окремої криптографічної проблеми, це розробка методів, що забезпечують конфіденційність разом з автентифікацією без використання узагальненої схеми. Починаючи з Джатла (англ. Jutla, Charanjit S) [9] і продовжуючи з Глігор (англ Virgil D. Gligor) та ін. [7], Роговей (англ. Rogaway Phillip) та ін. [15] з'явилися нові режими блочного шифрування, які поєднували конфіденційність та автентифікацію в єдиний, компактний режим. Такі схеми інтегрованого аутентифікованого шифрування (AE) обіцяли підвищення ефективності порівняно із загальним складом звичайних механізмів. Але схеми також мали значний (і спочатку непомітний) недолік: явна нездатність ефективно перевірити автентичність тексту асоційовних даних, наприклад заголовки пакетів чи інша статична інформація, прив'язуючи це до шифротексту.

Щоб побачити деякі проблеми, розглянемо такий протокол, який передає повідомлення $H\|C\|T$, де H - заголовок повідомлення, а C

визначається за допомогою шифрування простого тексту M під ключем K_1, а T визначається шляхом застосування MAC до рядка $H\|C$ ключем K_2. Припустимо, щоб зробити це швидше, хочемо модифікувати цей потік, використовуючи AE-схему, таку як OCB [15]. Не можна просто надіслати зашифрований OCB $H\|M$, оскільки, імовірно, H повинен був бути незмінним для цілей маршрутизації або розбору повідомлення. Також не можна OCB-шифрувати просто M та надсилати це разом із H, бо в цьому випадку б нічого не зроблено для автентифікації H. Не можна обійти цю проблему, надіславши повідомлення, що складається з H, а MAC застосований до H та зашифрованим OCB повідомлення M, тому що нічого б не було зроблено, щоб прив'язати H до M, але це подовжить передане повідомлення і витрачає час на обчислення MAC над інформацією, яка вже захищена автентичністю OCB. Можна було б надіслати зашифрований OCB $H\|M$ разом з H, тепер шифруючи H лише як засіб для забезпечення його автентичності, але це зробить ще раз подовження надісланого повідомлення. Можна спробувати стерти цю неефективність, якщо відправник опустить з шифротексту частину, що відповідає H (якщо припустити, що шифротекст має таку структуру, як це робиться з OCB). Але такий підхід, як правило, не працює: AE-схема не вимагає надання автентичності при неправильному використанні таким чином (а режими типу [15], [9] не надають автентичності, якщо їх неправильно використовувати). [14] виділяє AEAD як криптографічно значущу проблему та забезпечує їй обгрунтовану безпеку.

У цій роботі дано детальний огляд до схеми $AEAD$. Хоча перший і частина другого розділу є загальними в тому сенсі, що вони не покладаються на будь-які властивості схем $AEAD$ на основі суматорів, що дає можливість підготувати підгрунтя для представлення необхідності і мотивації існування таких схем - комбінація двох і більше схем $AEAD$, тобто з останнього твердження - ці схеми можливо масштабувати, що є безумовно необхідною властивістю, оскільки, якщо хоча б один з примітивів залишиться криптографічно стійким, то і вся схема буде

стійкою (див. [12])

Метою дослідження є аналіз швидкодії на 64 бітній архітектурі різних комбінації схем GCM, EAX, CCM з алгоритмом шифрування AES на основі 'чорної коробки', оскільки ці три схеми, AES присутні у безкоштовних криптографічних бібліотеках будь-якої популярної мови програмування і використання саме підходу 'чорної коробки' - не має необхідності втручатись у внутрішню роботу криптографічних бібліотек. Для дослідження швидкодії було обрано мову програмування *Java*. А тобто для досягнення мети необхідно вирішити такі завдання:

1) провести огляд опублікованих джерел за тематикою дослідження;

2) створити програму для аналізу швидкодії на обраному ЕОМ;

3) з отриманих результатів отримати графіки і зробити висновки про швидкодію комбінацій схем

Об'єктом дослідження є інформаційні процеси в системах криптографічного захисту. *Предметом дослідження* є властивості режимів автентифікованого шифрування у використанні з суматорами

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: методи абстрактної алгебри, теорії імовірностей, теорії кодування, теорії складності алгоритмів.

Наукова новизна отриманих результатів полягає у використанні результатів для побудови успішних атак по часу на схеми $AEAD$ на основі суматорів з внутрішнім станом.

Практичне значення результатів полягає в тому, що це відриває нам можливість тестувати комбіновані схеми з більшою кількістю параметрів.

1 ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Перевірка цілісності та автентичності інформації є першочерговою необхідністю в комп'ютерних системах та мережах. Зокрема, дві сторони, що спілкуються по незахищеному каналу, потребують метод, за допомогою якого інформація, що надсилається однією стороною, може бути перевірена на автентичність (немодифікована) [1] іншою стороною. Найчастіше такий механізм ґрунтується на секретному ключі, який поділяється між сторонами і застосовується код автентифікації повідомлення (MAC). (Інші терміни включають "перевірка на цілісність"або "криптографічна контрольна сума"). У цьому випадку, коли сторона A передає повідомлення стороні B, вона додає до повідомлення значення, яке називається тагом аутентифікації, обчислюється алгоритмом MAC, як функція від переданої інформації та спільного секретного ключа. При отриманні B обчислює таг аутентифікації для отриманого повідомлення за допомогою того ж механізму (і ключа) і перевіряє, чи отримане ним значення дорівнює тегу, приєднаному до отриманого повідомлення. Тільки якщо значення збігаються, то отримана інформація вважається такою, що не змінена на шляху від A до B. Мета - запобігти підробці, а саме - обчислення противником повідомлення (не надісланого законними сторонами) та відповідний дійсний тег аутентифікації. MAC найчастіше будуються з блокових шифрів типу DES. (Найпопулярнішим у цьому є $CBC - MAC$, проаналізований у [13], [2].) Проте останнім часом виник інтерес до ідеї побудови MAC з криптографічних хеш-функцій. Це особливо помітно в Інтернет-спільноті, де розробка протоколів безпеки призвела до необхідності у простих, ефективних та широко доступних механізмах MAC. Неважко зрозуміти, чому люди хочуть використовувати MAC з криптографічними хеш-функціями: популярні хеш-функції швидші, ніж блокові шифри в реалізації програмного забезпечення; ці реалізації

програмного забезпечення легко та вільно доступні; і більшість функцій не підпадають під дію патентного права. Більш складним питанням є те, як найкраще це зробити. Хеш-функції спочатку не були розроблені для використання для автентифікації повідомлень. (Однією з багатьох труднощів є те, що хеш-функції не є примітивними ключами, тобто не вміщують поняття секретного ключа.) Тому потрібно особливо обережно використовувати їх у використанні для цього. Зокрема, хоча було запропоновано багато конструкцій, їм не вистачає надійного та реалістичного аналізу безпеки. Таким чином, виникає потреба у конструкціях, які підтримують ефективність хеш-функцій, але підкріплюються більш жорстким аналізом їх безпеки. [1] Пропонують використовувати параметризовані геш-функції.

1.1 Параметризовані геш - фунції

Найбільш розповсюдженний підхід для параметризації хеш-функції - це введення ключа як частини хешованих даних функцією, наприклад, хешування даних x за допомогою ключа k виконується шляхом застосування хеш-функції F до $k\|x$. Підхід [1] полягає в тому, що замість використання фіксованого і відомого IV замінити його випадковим і секретним значенням, відомим лише сторонам. Це дозволяє краще моделювати параметризовані ключем хеш-функції, необхідні для аналізу безпеки цих функцій.

Використовуючи підхід параметризації IV,можна дати визначенням хеш-функції як сімейство функцій :

Означення 1.1. Нехай $l(n)$ деякий поліном. Тоді параметризованою геш-функцією назвемо :

$$h_k : \{0,1\}^{\leq l(n)} * K \to \{0,1\}^m,\ m \in \mathbb{N}, k \in K - \text{секрет}$$

Криптографічні вимоги до параметризованих геш-функцій

– Стійкість до пошуку колізій :

F - супротивник, що працює час $\leq T$:

$$F : K \to \{0,1\}^{\leq l(n)} * \{0,1\}^{\leq l(n)} \cup \{\varnothing\}$$

$\forall x : F_k(x) \neq x \Rightarrow \Pr_{k,F}\{F(k) \neq \varnothing\} \leq \epsilon$

– Стікість до пошуку прообразу (Pre)

F - супротивник, що працює ща час $\leq T$:

$$F : \{0,1\}^m \to \{0,1\}^{\leq l(n)} \cup \{\varnothing\} \Rightarrow \Pr_{x,F}\{h_k(F(h_k(x))) = h_k(x)\} \leq \epsilon$$

· alwaysPre : коли k обирається аналітиком, а $h_k(x)$ - довільне

· everywherePre : $h_k(x)$ - обирається аналітиком, а k - довільне

– Стійкість до пошуку другого прообразу (secPre).

F - супротивник, що працює час $\leq T$

$$F : \{0,1\}^{\leq l(n)} * K \to \{0,1\}^{\leq l(n)}$$

$\forall x : F_k(x) \neq x \ for \ time \leq T \Rightarrow \Pr\{h_k(F(x)) = h_k(x)\} \leq \epsilon$

· alwaysSec : коли k обирається аналітиком, а x - довільне

· everywhereSec : коли x обирається аналітиком, а k - довільне

В роботі [1] автори розширяють поняття стійкості до колізій на параметризовані геш-функції.

Означення 1.2. Сімейство параметризованих геш-функцій $\{F_k\} \in (\epsilon, t, q, L)$ - стійким до колізій, якщо будь-який противник, якому не надано ключ k, обмежений часом $Time \leq t$ і має можливість обчислити значення функції F_k на q повідомленнях $m_1, m_2, ..., m_q$ за своїм вибором, довжина кожного не більше L, не може знайти такі повідомлення $m \neq m^{'}$, для яких $F_k(m) = F_k(m^{'})$ з вірогідністю більше ϵ

1.2 Коди автентифікації повідомлень

Алгоритми кодів автентифікації повідомлень (MAC) є досить важливою складовою більшості мережевих протоколів. Вони забезпечують автентичність повідомлення між двома або більше учасниками.

Алгоритми MAC працюють майже в тому ж контексті, що і симетричні шифри. Вони приймають секретний ключ, який задає відображенння від повідомлення до власне MAC - фіксованої довжини рядок (тег). Хоча функції MAC приймають довілі великі повідомлення і дають вихід фіксованого розміру, вони не є еквівалентними хеш-функцій з точки зору безпеки.

Означення 1.3.

$$MAC_k : \{0,1\}^{\leq l(n)} * K \to \{0,1\}^m,\ m \in\ \mathbb{N}\ , k \in K - \text{ключ}$$

Криптографічні вимоги до MAC

– Ті ж вимоги, що і до параметризованої геш - функції

– Відновлення ключа (Key Recovery) : Необхідно, аби за множиною $(x_i, MAC_k(x_i))$ $i = 1...N$ знайти ключ k було складно.

– Підробка (Forgery) : Необхідно, аби за множиною $(x_i, MAC_k(x_i))$ $i = 1...N$ обчислити $MAC_k(x)$ $x \notin \{x_i\}$ було складно.

$MAC \in (\epsilon, T, q, q', q'')$ - зламний

Якщо $\forall$ супротивник який працює за час $\leq T$:

– Може обчислити MAC для q обраних повідомлень

– Може дізнатися MAC для q' відомих повідомлень

– Може перевірити q'' сформованих пар $(x, MAC_k(x))$ на коректність

$\Rightarrow$ підробка з ймовірністю $\leq \epsilon$

1.3 Автентифіковане шифрування

Означення 1.4. Автентифіковане шифрування

$$\Xi = \langle \mathcal{M}, K, C, T, E, D \rangle$$

де T - множина тегів (можливі значення MAC)

$$E : \mathcal{M} * K \to C * TD : C * T * K \to \mathcal{M} \cup \{\varnothing \text{ або } INVALID\}$$

Означення 1.5. Автентифіковане шифрування з асоційованими даними AEAD

$$\Xi = \langle \mathcal{M}, K, C, T, A, E, D \rangle$$

де A - асоційовані дані

$$E : \mathcal{M} * K * A \to C * T * A$$

$$D : C * T * K * A \to (\mathcal{M} \cup \{\varnothing \text{ або } INVALID\}) * A$$

Nonce схема. Нехай, A - супротивник має доступ до оракула шифрування $E^{\cdot\cdot}_K(\cdot)$. Цей оракул приймає (N, H, M) ($H \in A$ - асоційовані дані) повертає $E^{NH}_K(M) = CT$. Нехай $(N_1, H_1, M_1), ..., (N_q, H_q, M_q)$ позначимо запити до оракулу. Нехай, супротивник є nonce-схема, якщо $N_1, ..., N_q$ завжди різні, незалежно від відповідей оракулу.

Конфіденційність AEAD схем. Вважаємо, що $\forall$ супротивник, має доступ до оракула шифрування та nonce-схема. Перевага такого супротивника A у порушенні конфіденційності схеми $AEAD \prod = \langle E, D \rangle$, що має ключовий простір KEY це

$$\mathbf{Adv}^{priv}_{\Pi}(A) = \Pr\left[K \xleftarrow{\$} KEY : A^{E^{\cdot\cdot}_K(\cdot)} = 1\right] - \Pr\left[K \xleftarrow{\$} KEY : A^{\$^{\cdot\cdot}(\cdot)} = 1\right]$$

де $\$^{\cdot\cdot}(\cdot)$ позначимо оракул, який на запит (N, H, M) повертає випадкову

строку довжини $|M|$.

Автентифікація AEAD схем Надаємо супротивнику два оракули, оракул шифрування, а також оракул для верифікації $D_K^{\ddot{}}(\cdot)$. Останній оракул приймає на вхід (N, H, CT) і повертає 1, якщо $D_K^{NH}(CT) \in \mathcal{M}$ і повертає 0, якщо $D_K^{NH}(CT) = \varnothing$ або $INVALID$. Вважається, що супротивник повинен відповідати трьом умовам, і вони повинні виконуватись незалежно від відповідей на його запити до оракулу та незалежно від внутрішнього стану монет :

– Суперник A є nonce-схема. (Вважається, що ця умова стосується лише для оракула шифрування. Таким чином, запит, який використовується у запиті оракула шифрування, може бути використаний у запиті оракула для перевірки.)

– Супротивник ніколи не зробить запит до верифікуючого оракула (N, H, CT) : оракул шифрування раніше повернув CT у відповідь на запит (N, H, M).

– Супротивник A може зробити запит до верифікуючого оракулу тільки один раз. Тобто, він робить послідовність запитів до оракулу шифрування, потім запит до верифікуючого оракулу і зупиняється. Вважаємо, що такий супротивник виконав підробку, якщо його верифікуючий оракул повертає 1 у відповідь на запит, зроблений до нього. Перевага такого супротивника A у порушенні автентичності схеми $AEAD$, $\prod = \langle E, D \rangle$, що має ключовий простір KEY

$$\mathbf{Adv}_{\prod}^{auth}(A) = \Pr\left[K \xleftarrow{\$} KEY : A^{E_K^{\ddot{}}(\cdot), \hat{D}_K^{\ddot{}}(\cdot)} \text{ виконав підробку}\right]$$

IV схема шифрування Від nonce-схеми відрізняється тим, що приймає вектор ініціалізації $IV \in \{0,1\}^n$, $n \geq 1$

Конфіденційність IVE схеми : $IV \in_R \{0,1\}^n$. Нехай $\mathcal{E}^{\$}$ алгоритм шифрування, що повертає шифротекст і IV на основі випадково вибраного $IV \in_R \{0,1\}^n$.

Algorithm $\mathcal{E}_K^{\$}(M)$ // The probabilistic encryption scheme built from IVE scheme $\mathcal{E}$
$R \xleftarrow{\$} \{0,1\}^n$; $C \leftarrow \mathcal{E}_K^R(M)$; **return** $R \,\|\, C$

Визначимо перевагу супротивника A в зламі конфіденційності схеми $\prod$:

$$\mathbf{Adv}_{\prod}^{priv}(A) = \Pr\left[K \xleftarrow{\$} KEY : A^{\mathcal{E}_K^{\$}(\cdot)} = 1\right] - \Pr\left[K \xleftarrow{\$} KEY : A^{\$(\cdot)} = 1\right]$$

де $\$(\cdot)$ оракул повертає випадковий рядок довжини $n + |M|$, де M - вхідний рядок

Псевдовипадкові функції ПВФ. Сім'я псевдовипадкових функцій - це відображення $F : KEY \times D \to \{0,1\}^n$, D - непуста множина рядків, n розмір вихідних даних F. Позначимо F_K для функції $F(K,\cdot)$ і позначимо $K \xleftarrow{\$} KEY$, $f \leftarrow F_K$ як $f \xleftarrow{\$} F$. $\mathcal{R}_n^*$ безліч всіх функцій з $\{0,1\}$ на $0,1^n$; $\mathcal{R}_n^n$ набір усіх функцій з $0,1^n$ на $0,1^n$; і за $\mathcal{R}_n^{\mathcal{I}}$ набір усіх функцій з доменом $\mathcal{I}$ на $0,1^n$. Ідентифікуємо функцію з її ключем, роблячи $\mathcal{R}_n^n, \mathcal{R}_n^*$ і $\mathcal{R}_n^{\mathcal{I}}$ псевдовипадковими функціями. Перевага противника A у порушенні псевдовипадковості сімейства функцій $F : KEY \times \{0,1\}^* \to \{0,1\}^n$

$$\mathbf{Adv}_F = \Pr\left[K \xleftarrow{\$} KEY : A^{F_K(\cdot)} = 1\right] - \Pr\left[\rho \xleftarrow{\$} \mathcal{R}_n^* : A^{\rho(\cdot)} = 1\right]$$

Сім'я функцій $E : KEY \times D \to \{0,1\}^n$ блочний шифр, якщо $D = \{0,1\}^n$ і кожен E_K - деяка перестановка. Нехай, $\mathcal{P}_n$ всі можливі перестановки на $\{0,1\}^n$. І визначимо

$$\mathbf{Adv}_E^{prp}(A) = \Pr\left[K \xleftarrow{\$} KEY : A^{E_K(\cdot)} = 1\right] - \Pr\left[\pi \xleftarrow{\$} \mathcal{P}_n : A^{\pi(\cdot)} = 1\right]$$

Використання ресурсів. Нехай xxx - позначення переваги : $\mathbf{Adv}_{\prod}^{xxx}(A)$; Позначимо $\mathbf{Adv}_{\prod}^{xxx}(R)$ як максимальне значення $\mathbf{Adv}_{\prod}^{xxx}(A)$ над усіма супротивниками A, які використовують не більше R ресурсів. Під час підрахунку використання ресурсів супротивник, максимізує над

усіма можливими відповідями оракул, включаючи ті, які не вдалося повернути жодним експериментом. Ресурси, що нас інтересують, є : t - час роботи; q - загальна кількість запитів оракулу; q_e - кількість запитів до першого оракула; q_v - кількість запитів до другого оракула; і σ - складність даних, тобто кількість пам'яті, що використовуємо. Час виконання алгоритму t - його фактичний час роботи (щодо деякої фіксованої моделі обчислення RAM) плюс розмір його опису (щодо деякого стандартного кодування алгоритмів). Складність даних σ визначається як сума довжин усіх рядків, закодованих у запитах оракула супротивника, плюс загальна кількість всіх цих рядків. Виміряємо довжини рядків в n-бітових блоках. Кількість блоків у рядку M визначається як $\|M\|_n = max\{1, \lceil |M|/n \rceil\}$, отже пустий рядок має 1 блок. Отже, кількість доступних ресурсів позначаємо, як (t, q, σ). Зокрема, $Time_F(\sigma)$ - максимальна кількість часу, аби обчислити функцію F на входах загальної довжини σ. $Time_{\mathcal{E}}(\sigma)$ - час обчислення випадкового елемента $K \xleftarrow{\$} KEY$ + максимальна кількість часу для обчислення $\mathcal{E}_K$ на аргументах загальної довжини σ.

Захищені канали зв'язку Для того, щоб стверджувати про будь-які результати, тобто, що певна комбінація шифрування та аутентифікації забезпечує безпечну комунікацію, нам потрібно визначити, що розуміється під таким «захищеним зв'язком». Для цього використовуємо модель захищених каналів, запроваджену Канетті та Кравчиком [6], [10], яка покликана фіксувати стандартну практику безпеки мережі, в якій комунікація захищена через сеанси між парами комунікаційних сторін, і де кожен сеанс складається з двох етапів. По-перше, дві сторони запускають протокол обміну ключами, який встановлює автентифікований та секретний ключ сеансу, який спільно використовується між сторонами. Потім, на другому етапі, цей ключ сеансу використовується разом із криптографічними функціями симетричного ключа для захисту цілісності та / або секретності переданих даних. Формалізм [6] передбачає визначення протоколу обміну

ключами для впровадження сеансу та ключового етапу встановлення, а також двох функцій, *snd* та *rcv*, які визначають дії, застосовані до переданих даних через незахищене середовище передачі даних. Протокол, що слідує за цим формалізмом, називається в [6] «протоколом мережевих каналів», а його безпека визначається з точки зору автентичності та секретності. Ці поняття визначені в [6] в контексті комунікацій, контрольованих зловмисником, з повним контролем інформації, що надсилається, і з можливістю підробки \сеансів і сторін. Функція *snd* представляє операції та перетворення, застосовані до повідомлення його відправником, щоб захистити його від дій супротивника (нападника). А саме, коли повідомлення m має передаватися від сторони P до сторони Q в рамках сеансу, встановленого між цими сторонами, функція *snd* застосовується до m і, можливо, до додаткової інформації, такої як ідентифікатор повідомлення. Визначення *snd* зазвичай складається із застосування деякої комбінації MAC та симетричного шифрування, введеного через ключ сеансу. Функція *rcv* описує дію на приймальному кінці для розшифрування та верифікації вхідних повідомлень. Тобто, будь-яка модифікація повідомлень, отриманих зловмисником по комунікаційним посиланням, включаючи введення або повтор повідомлення, повинна бути виявлена та відхилена одержувачем. Секретність оформлена в традиції семантичної безпеки: серед безлічі повідомлень, обмінюваних сеансом, зловмисник вибирає пару тестових повідомлень, з яких надсилається лише одне; мета нападника - здогадатися, яке саме було надіслано. Захищеність вважається, що дотримана, якщо зловмисник не може здогадатися з вірогідністю, значно більшою від 1/2. Протокол мережевих каналів називається протоколом захищених каналів, якщо він досягає як автентифікації, так і секретності у тому сенсі, який викладений вище.

Узагальнені схеми побудови E_k , MAC_k , $k \to (k_e, k_m)$

- *Encrypt & MAC* : $\bar{\mathcal{E}}(K_e, M) = \mathcal{E}(K_e \| M) \| \mathcal{T}(K_m, M)$.
- $MAC \to$ *Encrypt* : $\bar{\mathcal{E}}(K_e, M) = \mathcal{E}(K_e \| \mathcal{T}(K_m, M))$.

– *Encrypt* → $MAC : \bar{\mathcal{E}}(K_e, M) = C \| \mathcal{T}(K_m, M)$, де $C = \mathcal{E}(K_e, M)$

$\bar{\mathcal{E}}$ - алгоритм шифрування з автентифікацією AE.

Висновки до розділу 1

Перший розділ дав нам основні поняття, які будуть використовуватись для проведення детального огляду режимів автентифікованого шифрування в наступному розділі.

2 МЕТОДИ ПОБУДОВИ АВТЕНТИФІКОВАНОГО ШИФРУВАННЯ

2.1 Аналіз узагальнених схем побудови

Розглянемо криптографічну стійкість схем $Encrypt \rightarrow MAC(EtA)$, $MAC \rightarrow Encrypt(MtE)$, $Encrypt\&MAC(E\&M)$ узагальненої схеми побудови з алгоритмами симетричного блочного шифрування та функціями MAC, де єдине припущення полягає в тому, що E - алгоритм шифрування є стійким у сенсі $IND - CPA$ і MAC стійкий до атак на основі вибраного тексту.

Означення 2.1. Криптосистема є стійкою у сенсі $IND - CPA$, якщо будь-який супротивник за поліноміальну кількість часу має лише незначну перевагу у грі, що вказана нижче, і виграє її з ймовірністю $\frac{1}{2} + \epsilon(k)$, де $\epsilon(k)$ знехтувально мала функція від параметру безпеки k, тобто $\forall\ poly()\exists\ k_0 : |\epsilon(k)| < \left|\frac{1}{poly(k)}\right| \forall\ k_0 > k_0$

Супротивник моделюється ймовірнісною машиною Тьюрінга, що означає, що він повинен завершити гру і видати здогадку з поліноміальну кількість кроків. У цьому визначенні $E(PK, M)$ являє собою шифрування повідомлення M ключем PK :

– Коористувач генерує пару ключів PK, SK на основі деякого параметра безпеки k (наприклад, розмір ключа в бітах) і публікує PK противнику. Користувач зберігає SK.

– Супротивник може виконувати будь-яку кількість шифрувань або інших операцій.

– Зрештою, супротивник подає користувачу два різних обраних повідомлення M_0, M_1.

– Користувач обирає біт $b \in \{0, 1\}$ рівномірно і одночасно відправляє шифротекст $CT = E(PK, M_b)$ назад супротивнику.

– Супротивник вільний виконувати будь-яку кількість додаткових

шифрувань чи обчислень.

– I видає здогадку для обраного b

Наша увага зосереджена на доцільності цих методів для забезпечення безпеки переданих даних у реалістичній обстановці мереж, що контролюються супротивником. Іншими словами, нас цікавить, чи кожен із цих методів при застосуванні до каналів зв'язку, досягає конфіденційності та цілісності інформації. Як було показано в роботі [10] побачимо, що лише схема EtM є найбільш захищеної, стійкою порівняно з двома іншими.

Encrypt − then − Mac схема

Вказані результати взяті з робіт, [6], [10], [3] коротко представимо для повноти.

Теорема 2.1. *[6] Якщо ENC є симетричною схемою шифрування, захищеною в сенсі $IND-CPA$, а MAC є стійким до підробок, то схема $EtA\langle ENC, MAC\rangle$ реалізує захищений канал.*

З термінології розділу 1, сенс вищевказаної теореми полягає в тому, що якщо в моделі мережевих каналів [6] застосовується до кожного переданого повідомлення функція $EtA\langle ENC, MAC\rangle$ (як функція snd), то конфіденційність і автентифікація результативних мережевих каналів гарантована. Точніше, доводячи вищезгадану теорему, в [6] задано функцію snd наступним чином. По-перше, з кожного ключа сеансу виводяться пари (обчислювально незалежних) ключів, k_a та k_e. Потім для кожного переданого повідомлення m вибирається унікальний ідентифікатор повідомлення $m-id$ (наприклад, номер послідовності). Нарешті, функція snd виробляє трійку (x, y, z), де $x = m - id$, $y = ENC_{k_e}(m)$, $z = MAC_a(m - id, y)$. У вхідному повідомленні (x', y', z') функція rcv перевіряє унікальність ідентифікатора повідомлення x' та дійсність тегу $MAC\ \ z$ (обчислюється на (x', y')); якщо перевірки досягли успіху, y' розшифровується ключем k_e, а отриманий текст приймається як дійсне повідомлення. Основний внесок [10] полягає в тому, що він показує, що результат, як у теоремі 2.1, не працює з іншими двома схемами -

MtE та $E\&M$ (навіть якщо використовувані ключі розподіляються безпечно). Отже, будь-який протокол захищених каналів, призначений для роботи з будь-якою комбінацією шифрування стійкого у сенсі проти атак до вибраного тексту та стійкого до підробок MAC, повинен використовувати метод EtM. Однак у [10] вищевказану теорему можна розширити для схеми MtE, якщо надати більш сильної властивості шифрування; зокрема, [10] показує два важливі випадки, які задовольняють додаткову вимогу захищеності.

Mac – then – Encrypt схема

MtE не є захищеною в загальному випадку. В роботі [6] було показано, що схема $MtE\langle ENC, MAC\rangle$ не є гарантовано захищеною для реалізації захищених каналів, навіть якщо ENC є $IND-CPA$ і MAC забезпечує захищеність повідомлень від атак обраного тексту. Спочатку, однак, коротко обговоримо, чому цей результат не випливає з [3], де показано, що композиція MtE (розглядається як схема шифрування) не обов'язково забезпечує $IND-CCA$. В [6] продемонстровано, що $IND-CCA$ не є необхідною умовою комбінації шифрування та MAC для реалізації захищених каналів. Приклад наводить основна конструкція захищених каналів у [6] (Теорема 2.1): якщо MAC, використовуваний у цій схемі, задовольняє умову регулярності безпеки MAC, а не посиленим поняттям, описаним в останньому зауваженні розділу **??**, то ця конструкція гарантує безпечні канали, але не обов'язково захищеність у сенсі CCA. (Наприклад, якщо функція MAC має властивість, що при зміні останнього біта тегу аутентифікації не змінює валідність тегу, то схема в [6] не є захищеною у сенсі $IND-CCA$, але її достатньо для реалізації захищених каналів. Таким чином, якщо хочемо встановити незахищеність каналів аутентифікації та зашифрування під загальним складом, потрібно показати чіткий приклад та успішну атаку.У цьому прикладі схема шифрування є $IND-CPA$ (тобто задовольняє умову захищеності по Шеннону), але в поєднанні з будь-якою функцією MAC за методом MtE секретність схеми повністю руйнується під дією атаки :

Функція шифрування ENC. Почнемо з визначення схеми шифрування ENC, яка може базуватися на будь-якому потоковому шифрі ENC (тобто будь-якій функції шифрування, яка використовує випадковий або псевдовипадковий зсув і операцію XOR з даними). Схема ENC^* зберігає $IND-CPA$ основної схеми ENC. Зокрема, якщо ENC має досконалу захищеність (тобто використовує ідеальне одноразове шифрування зсувом), це робить і ENC^*. Далі визначаємо ENC^*.
Дано n-бітний текст x, ENC спочатку застосовує кодування x у $2n$-бітний рядок $x^{'}$, отриманий шляхом представлення кожного біта $x_i, i = 1..n$, в x двома бітами в $x^{'}$ наступним чином:

– якщо $x_i = 0 \ \rightarrow \ (0,0)$

– $x_i = 1 \ \rightarrow \ (0,1)$ або $(1,0)$ довільним вибором

Потім функція шифрування ENC застосовується до $x\prime$. Для дешифрування $y = ENC^*(x)$ спочатку застосовується функція дешифрування ENC для отримання $x^{'}$, а потім розшифровується в x шляхом відображення пари $(0,0) \rightarrow 0$ і будь-якої пари $(0,1)$ або $(1,0) \rightarrow 1$. Якщо $x^{'}$ містить пару $(1,1)$, то декодування виводить знак $INVALID$ або $\perp$.

Атака, коли використовується лише шифрування. Припустимо, що до переданих даних застосовується лише ENC^* (тоді ми розглянемо випадок MtE, коли до даних застосовано MAC перед шифруванням).

У цьому випадку, коли зловмисник A бачить переданий шифротекст $y = ENC(x)$, він може здогадатись про перший біт x_1 з x наступним чином. Він перехоплює y, міняє ($0 \leftrightarrow 1$ і) перші два біти (y_1, y_2 і посилає модифікований шифротекст $y^{'}$ отримувачу. Якщо A може отримати інформацію про те, виводить дешифрування дійсний простий текст чи $\perp$, то А здогадався перший біт x. Це так, оскільки, як це легко видно, модифікований $y^{'}$ є дійсним тоді і лише тоді, коли $x_1 = 1$. (ми використовуємо потоковий алгоритм шифрування $x^{'}$) Це порушує секретність каналу. (*Зауваження*, описана атака може бути застосована до будь-якого біта простого тексту). Одне питання, що виникає, - чи

реально припустити, що зловмисник дізнається валідність шифротексту. Відповідь полягає в тому, що це так для багатьох практичних програм, які покажуть помітну зміну поведінки, якщо шифротекст недійсний (зокрема, багато додатків повернуть повідомлення про помилку в цьому випадку).

Розглянемо протокол, який передає паролі та використовує ENC^* для захисту паролів в мережі (це, наприклад, одне з найпоширеніших застосувань SSL). Наведена вище атака, якщо застосована до одного з бітів пароля (ми припускаємо, що зловмисник знає розміщення поля пароля у переданих даних) буде працювати наступним чином. Якщо атакований біт дорівнює 1, то автентифікація пароля буде успішною, незважаючи на зміну шифротексту. Якщо це 0, то автентифікація пароля не вдасться. У цьому випадку про успіх або провал повідомляється назад на віддаленій машині, а потім дізнається зловмисник. У додатках, де один і той же пароль використовується декілька разів (знову ж таки, як у багатьох програмах, захищених SSL), зловмисник може відновити пароль біт-за-бітом. Те ж саме може бути застосовано до іншої конфіденційної інформації, наприклад до номерів кредитних карток, де помилка в номері зазвичай повідомляється назад, а інформація про дійсність / недійсність дізнається А. *Атака на схему* $MtE\langle ENC^*, MAC\rangle$ Розглянемо випадок, коли шифрування застосовується не лише до даних, але й до функції MAC, обчисленої на цих даних. Чи застосовується вищевказана атака? Відповідь - так. MAC застосовується до даних перед кодуванням та шифруванням, тому, якщо вихідний біт дорівнює 1, зміна шифротексту призведе до того ж розшифрованого простого тексту, і тоді перевірка MAC буде успішною. Аналогічно, якщо початковий біт дорівнює 0, а розшифрований простий текст матиме 1, то $MAC \rightarrow INVALID$. Тепер зловмисникам потрібно лише інформація про те, чи валідний MAC чи ні. Зауважте, що в деякому сенсі MAC просто погіршує ситуацію, оскільки незалежно від семантики програми зловмиснику простіше дізнатись про помилку аутентифікації: або через повернені повідомлення про помилку,

або за допомогою інших ефектів на програму, які можуть спостерігати нападники. Отже, приклад, що використовує $ENC^{\$}$, безумовно, достатній для того, щоб показати, що схема MtE може бути незахищеною, навіть якщо функція шифрування стійка у сенсі $IND - CPA$ та MAC стійкий до підробок (зауважте, що цей висновок не залежить від конкретної формалізації захищених комунікацій; будь-якого розумного визначення безпеки повинен позначати вищезазначений протокол як небезпечний). Тому, якщо потрібно проаналізувати на стійкість $MtE\,\langle ENC, MAC\rangle$ для конкретних функцій ENC та MAC, потрібно проаналізувати комбінацію в цілому або використовувати більш сильні або специфічні властивості функції шифрування. Цікавим питанням є те, наскільки правдоподібним є те, що люди коли-небудь використовуватимуть схему шифрування, наприклад ENC^*. Зауважимо, що хоча ця схема не є найбільш природним механізмом шифрування, деякі (однаково небезпечні) варіанти її можуть виникнути на практиці. По-перше, застосування кодування до простого тексту перед шифруванням багато разів використовується для заміщення та інших цілей і є особливо поширеною практикою в алгоритмах шифрування відкритих ключів. По-друге, кодування такого типу може мотивуватися більш високими вимогами до безпеки: напр. щоб запобігти зловмиснику дізнатися точну довжину переданих повідомлень або іншої інформації про аналіз трафіку. У цьому випадку можна використовувати кодування, подібне до ENC^*, але з кодами змінних розмірів. (Просто для того, щоб зазначити: зауважте, що в наведених вище прикладах є хороший приклад аналізу трафіку, коли зловмисник має багато чого здогадуватись з повідомлень про помилки; навіть у випадках, коли ця інформація зашифрована, її зазвичай можна дізнатися за допомогою аналізу довжини пакетів і т. д.) Ще одна конфігурація, коли вводиться кодування простого тексту для підвищення безпеки, - це боротьба з атаками часу та аналізу потужності. Суть полягає в тому, що вкрай бажано мати схеми, стійкі до загального складу та не вразливі, коли в алгоритм вносяться, здавалося б, нешкідливі зміни

(або коли приймається новий більш стійкий або більш ефективний алгоритм чи режим)

Теорема 2.2. *[6] Нехай ENC - функція шифрування $IND-CPA$, а MAC - стійкий до підробок у сенсі атак на основі вибраного тексту. Якщо функція $MtE\langle ENC, MAC\rangle$, розглянута як схема шифрування, є $CUF-CPA$, то $MtE\langle ENC, MAC\rangle$ реалізує захищені канали.*

Теорема 2.3. *[10] Нехай MAC - є стійким до атак одного запиту, тоді $MtE\langle OTP_{\$}, MAC\rangle$ є $CUF-CPA$ (і тоді за Теоремою 2.2 реалізує захищені канали зв'язку). Більш точно, для будь-якого супротивника $\mathcal{F}$, що атакує схему $MtE\langle OTP_{\$}, MAC\rangle$ є $CUF-CPA$, і потребує T часу виконає підробку з ймовірністю $\mathcal{E}_U \leq q^2 \cdot 2^{-l} + \mathcal{E}_M(1, p, T')$, де l параметр $OTP_{\$}$, q кількість запитів $\mathcal{F}$ виконаних за час атаки, p - верхня межа довжини кожного з таких запитів і довжини результату підробки, $T' = T + cqp, c \equiv const$,*

Mac – and – Encrypt

M&E схема не є захищеною в загалі кажучи. [10] Слід завуважити, що навіть за стійкості MAC до підробок ця схема не може гарантувати захищеності (навіть проти пасивного підслуховувача). Це так, оскільки MAC може бути захищений від підробок, але взаємна інформація між результатом автентифікації і текстом може бути не нульова $\mathcal{I}(M, MAC_{k_a}(M)) = H(M) - H(M|MAC_{k_a}(M)) \neq 0$. Таким чином, справді цікавим питанням є те, чи стане схема більш захищеною, якщо ми уникнемо цієї очевидної слабкості за допомогою використання MAC, такого, який реалізується за допомогою псевдовипадкової функції, або коли тег MAC зашифрований (вважається,що більшість, якщо не всі, функції MAC, використовувані на практиці, стійкі до підробок). На жаль, однак, напад з попереднього розділу застосовується і тут, таким чином показано незахищеність загалом схеми EM навіть у вищезазначених сильніших формах MAC.

2.2 Блочні режими автентифікованого шифрування

Режим GCM

Стандарт NIST SP 800-38D Galois Counter Mode (GCM) - це режим роботи блокового шифру, який використовує універсальну геш-функцію $GHASH_H(M)$ в GF(2^m) для забезпечення автентифікованого шифрування. Він може бути реалізований апаратно для досягнення найбільш ефективного використання апаратних ресурсів.

Вхідні та вихідні дані GCM має дві операції, автентифіковане шифрування та автентифіковане розшифрування. Автентифіковане шифрування має чотири входи, кожен з яких є бітовим рядком:

• Потаємний ключ K, довжина якого відповідає шифру основного блоку.

• Вектор ініціалізації IV, який може мати будь-яку кількість біт між 1 і 2 64. Для фіксованого значення ключа, кожне значення IV повинно бути виразним, але не мати однакових довжин. 96-розрядні Значення IV можна обробити більш ефективно, так що довжина рекомендується застосовувати в ситуаціях яка ефективність є критичною.

• Кількість бітів цільового тексту M : $|M| \in \left[0, 2^{39} - 256\right]$

• Приєдані автентифіковані дані (Additional Authenticated Data), позначаємо A.

Вихідні дані :

• Шифротекст C : $|M| = |C| \in \left[0, 2^{39} - 256\right]$

• T - тег аутентифікації, MAC. $t = |T| \in [0, 128]$ *Шифрування*

Нехай $m = 128$ - степінь розширення поля $\mathbf{GF}(2^{128})$, n - кількість блоків, на які ми розбиваємо наше повідомлення, тобто

$$X_1, X_2, ...X_t^* \leftarrow X \ : \ |X_k| = m, \ k = 1..t$$

X_t^* - доповнений до розміру m блок X_t, елемент $H \in GF(2^m)$ таке, що

$$GHASH_H(X) = \bigoplus_{i=1}^{t} x_i \cdot H^{t-i+1}$$

$H = E_K(0^m)$ - ключ для гешування, IV - nonce

$$Y_0 = \begin{cases} IV \parallel 0^{31}1 & \text{якщо } |IV| = 96 \\ GHASH_H(IV||0^{m/2}|| < len(IV) >) & \text{інакше} \end{cases}$$

$Y_i = inc_{32}(Y_{i-1})\ i = 1, .., n$

$C_i = P_i \oplus E_K(Y_i) \quad \text{for } i = 1, ..., n-1$

$C_n = P_n \oplus MSB_u(E_K(Y_i))$

$T = MSB_t(GHASH_H(A, C) \oplus E_K(Y_0))$

$GHASH_H(A, C) = X_{m+n+1}$

X_i, $i = 0\ ..\ m+n+1$ таке, що

$$X_i = \begin{cases} 0 & \text{,якщо } i = 0 \\ (X_{i-1} \oplus A_i) \cdot H & i = 1..m-1 \\ \left(X_{m-1} \oplus \left(A_m^* \parallel 0^{128-v}\right)\right) \cdot H & i = m \\ (X_{i-1} \oplus C_i) \cdot H & i = m+1..m+n-1 \\ \left(X_{m+n-1} \oplus \left(C_m^* \parallel 0^{128-u}\right)\right) \cdot H & i = m+n \\ (X_{m+n} \oplus (len(A) \parallel len(C))) \cdot H & i = m+n+1 \end{cases}$$

Розшифрування

$H = E_K(0^m)$, IV - nonce

$$Y_0 = \begin{cases} IV \parallel 0^{31}1 & \text{якщо } |IV| = 96 \\ GHASH_H(\{\}, IV) & \text{інакше} \end{cases}$$

$$T' = MSB_t(GHASH_H(A, C) \oplus E_K(Y_0))$$

$$Y_i = incr_{32}(Y_{i-1}) \quad \text{i} = 1..n$$

$$P_i = C_i \oplus E_K(Y_i) \quad \text{i} = 1..n$$

$$P_n^* = C_n^* \oplus MSB_u(E_K(Y_n))$$

Тег T', який обчислюється операцією дешифрування, порівнюється з тегом T, який отримує інша сторона, разом з шифротекстом C. Якщо два $T \neq T'$, то повертається спеціальний символ, що означає невдачу, наприклад $FAIL$, $\varnothing$.

Зауваження

– $E_K(Y_0)$ *повинен бути одноразовий (nonce)*

– *Обмежено кількість блоків повідомлення*

– *Якщо E - ідеальний шифр, тобто моделює випадкову перестановку, то GCM - доказово стіка схема до підробок*

– *GCM був прийнятий NIST для використання разом з AES.*

GHASH

GHASH належить до класу поліноміальних кодів автентифікації повідомлень Вегмана - Картера (Wegman - Carter). Спочатку були запропоновані незалежно трьома авторами [4, 5, 6]. Нижня межа стійкості до підробок подібного n бітного кода автентифікації $2^{\frac{-n}{2}}$. У роботі [7] було надано докази, що це й верхня межа.

У GHASH всі операції відбуваються у $GF(2^{128})$ над многочленом $f(x) = x^{128} + x^7 + x^2 + x + 1$ *Циклічні атаки на GHASH* З властивості мультиплікативної групи $GF(2^{128})$, $H \in GF(2^{128}) : \exists\ n = ord(H)$, що $H^{k=n\cdot m+r} = H^r$. З чого слідує, якщо ми зможемо поміняти місцями X^k, X^r і результат аутентифікації, наш таг, не зміниться, то це означає, що і є підробкою.

Властивість мультиплікативної групи $GF(2^{128})$ дає нам можливість це зробити для довільних $k\ r : k - r \equiv 0\ mod\ n$

З теореми Лангранжа, нам відомо, що $\forall H :\ H \subseteq G\ \ |H| \vdots |G|$, де G - скінченна група, а H - підгруппа G. Тому

Нехай, надалі вважаємо $\mathcal{G}$ мультиплікативною групою поля $GF(2^{128})$, якщо інше не вказано. Ми можемо легко факторизувати, як число Ферма,

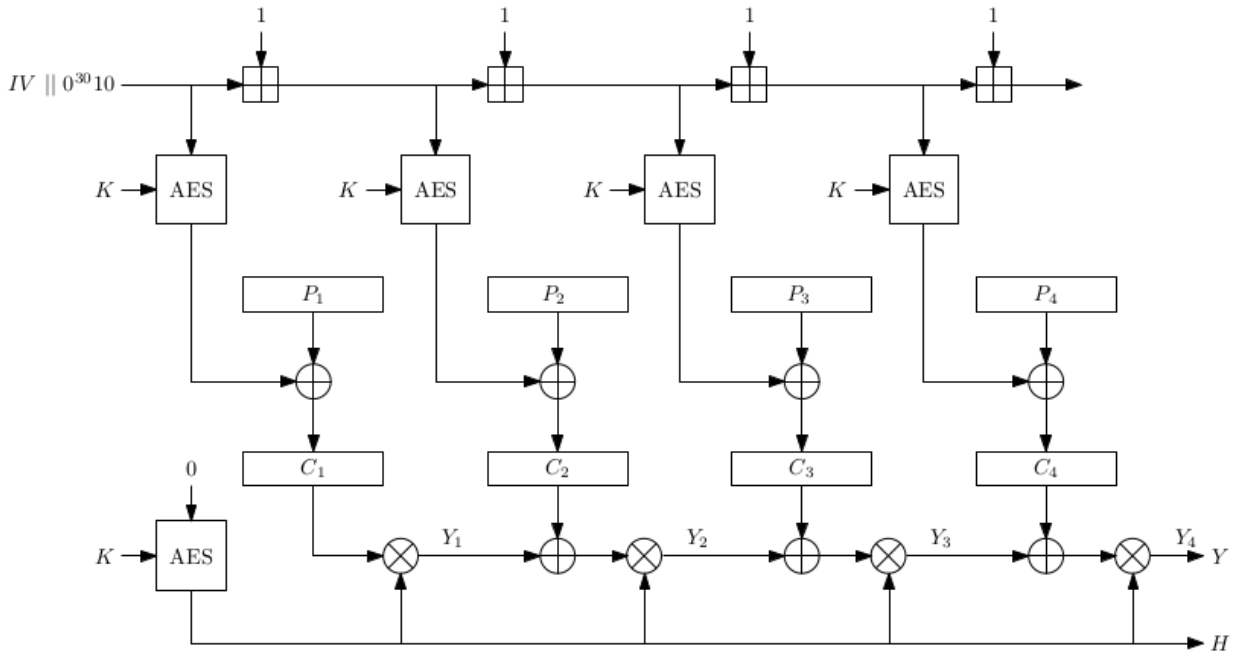
$$2^{128} - 1 = 3 \cdot 5 \cdot 17 \cdot 257 \cdot 641 \cdot 65537 \cdot 274177 \cdot 6700417 \cdot 67280421310721$$

А отже, існує $2^9 = 512$ підгруп групи $\mathcal{G}$

Підробка GHASH Зловмисник (супротивник, adversary) за атаки підробки GHASH не знає H, тому буде виконувати підробку міняючи місцями випадкові два або більше блоків.

Теорема 2.4. *Нехай* $2^{128} - 1 \vdots n$. *Тоді ймовірність успішної підробки повідомлення, міняючи місцями блоки* X_k, X_l, $k \equiv l\ mod\ n$, *не менше* $\frac{n+1}{2^{128}}$

Доведення. Якщо $2^{128} - 1 \vdots n$ тоді це означає, що n один з 512 дільників мультиплікативної групи нашого поля, тобто є розмірністю однієї з 512 підгруп. Тоді, якщо $n \vdots ord(H)$ тоді міняємо місцями $H^k \equiv H^l$ і підробка успішна. Тоді, оскільки існує n елементів підрупи, кожен цикл



$\bigotimes$ операція множення, $\bigoplus$ операція додавання в $GF(2^{128})$

Рисунок 2.1 – Схема GCM для тексту довжиною 4 блоки

єдиний і включаємо ймовірність $H = 0$, то ймовірність успішної підробки $\frac{n+1}{2^{128}}$ □ *Зауваження*

– *Якщо твердження теореми не виконується, тобто* n *не ділить потужність* $\mathcal{G}$*, немає причин очікувати, що ймовірність успішною підробки більше* $\frac{1}{2^{128}}$

– *Якщо було виконано успішну підробку, то ми зможемо виконати й довільну кількість успішних послідовних підробок з ймовірність* 1*, якщо* nonce *або* IV *не було змінено*

Таким чином, якщо $ord(H) \mid (k - l)$ таг автентифікації залишається коректним, оскільки $X_i \cdot H^{m-k+1} + X_l \cdot H^{m-l+1} \equiv const = t$ і $H^{m-k+1} \equiv H^{m-l+1} \equiv H_t \ \rightarrow \ X_k + X_l \equiv t \cdot H_t^{-1}$

Отож, ми можемо знайти слабкі H досить легко, але нам звісно необхідно знайти K, з яких ми і отримуємо H. Аби визначити груповий порядок ми можемо використати алгоритм, подібний до алгоритму Сільвера-Поліга-Хелмана. Тут, використовуючи факт про :

Теорема 2.5. *Нехай* p *простий* $2^{128} - 1 \vdots p$*. Тоді* $ord(H) \vdots p \Leftrightarrow$

$$H^{\frac{2^{128}-1}{p}} \neq 1$$

Доведення.

$\Rightarrow$

$ord(H) \vdots p \ \rightarrow \ ord(H) = p \cdot k$, де $k \in \mathbb{Z} \ \rightarrow H^{\frac{2^1 28-1}{p}} \equiv H^{\frac{ord(H)\cdot t}{p}} \equiv H^{k*t} \not\equiv 1$

$\Leftarrow$

$H^{\frac{2^{128}-1}{p}} \not\equiv 1$, то оскільки $2^{128} - 1 \vdots p$, маємо $H^k \not\equiv 1, \ k \in \mathbb{Z}$, з чого слідує, що або $ord(H) = p$ або $ord(H) \vdots p$ □

Ми знаємо розклад на множники числа $2^{128} - 1$, а отже пробігаючи кожний p простий дільник потужності групи $\mathcal{G}$ застосовуємо попердню теорему ми отримаємо порядок $H \in \mathcal{G}$ Зокрема нам знадобиться ефективний алгоритм піднесення елемента до степеня двійки. Можемо застосувати алгоритм Горнера. Також можна використати ефективний алгоритм для нашого випадку, описаний в роботі [7].

Пришвидшення підробки методами лінійної і абстрактної алгебри

Піднесення до степеня числа Ферма $2^{2^n} + 1$ для елемента групи $\mathcal{G}$ можна ефективно обчислювати за допомогою факту, що для $\mathbf{GF(2^n)}$
$\exists!$ матриця $2^n \times 2^n$ бітова матриця $\mathcal{L}$

$$\forall X \in \mathcal{V}(2^n) \ : \ X^2 = \mathcal{L}X$$

З елементами поля $\mathbf{GF(2^n)}$ працюємо, як з бітовими векторами.

Якщо піднесемо до квадрату k раз $\mathcal{L}$, то $\mathcal{L}^k$:

$$X^{2^{2^k}} = \mathcal{L}^k X$$

Тоді, з передобчисленням ми можемо отримати :

$$H^{F_n} = \mathcal{L}^n H \times H$$

$2^{128} - 1 = \prod_{i=0}^{6} F_i$. Перевірити чи ділиться порядок елемента H на деяке

число Ферма зводиться до перемноження матриц і елементу. Наприклад, нам необхідно перевірити, що порядок $ord(H)$ ділить число Ферма F_4. Маємо :

$$\mathcal{L}^6(\mathcal{L}^5(\mathcal{L}^3(\mathcal{L}^2(\mathcal{L}^1(\mathcal{L}H \times H) \times H) \times H) \times H) \times H) = 1$$

Тож, існування циклічних атак до $GHASH$ проявляються через можливість факторизації потужності групи $\mathbf{G}$. Що ми можемо зробити в такому випадку? Досить очевидне рішення це виконувати операції в полі $GF(2^{127})$, звідси $2^{127} - 1$ відоме нам просте число Мерсена.
В роботі [16] рекомендується використовувати числа Софі Жермен, тобто числа виду $q = \frac{p-1}{2}$ - просте. Наприклад $GF(2^{128} + 12541)$. Однак це значно сповільнить час обчислення тагу автентифікації. Зокрема, за допомогою розширених інструкцій процесора SIMD ми можемо виконувати паралельні обчислення. *Швидке обчислення GHASH* Оскільки кількість блоків t, на яке ми розбиваємо наше повідомлення X може бути більше, ніж $m = 128$. Виникає необхідність в тому, аби зменшити степінь поліному $GHASH(\cdot)$, отримати залежність

$$H^t = f(H^{t-1}, H^{t-2}, H^{t-3}, .., H)$$

З алгебри нам відоме поняття мінімального поліному g_H для
$H \in \mathbf{GF}(2^{128})$, що $m \vdots deg(g_H) = d$ такий, що $g_H(H) = 0$ і поняття характеристичного поліному :

$$f_H(x) = (\prod_{i=0}^{d-1}(x - H^{2^i}))^{\frac{m}{d}} = g^{\frac{m}{d}}$$

Маємо d різних коренів. Представимо $f_H(x) = \sum_{k=0}^{m} a_k \cdot x^k$. Досить очевидно, що $a_m = 1$. Зокрема $a_0 = 1$, оскільки, інакше б протиріччя з g - мінімальний поліном.

Для $m = 128$ маємо :
$f_H(H) = 0 \to \sum_{k=0}^{m} a_k \cdot H^k = 0 \to H^{128} = \sum_{k=0}^{127} a_k H^k$

Якщо ви використовуєте $\mathbf{GF}(2^{128})$, то в роботі [11] для цього вже приведені необхідні рівняння, які дають можливість виконати паралельні обчислення $GHASH$ для довгих повідомлень. У разі використання, наприклад $\mathbf{GF}(2^{127})$, що дає можливість захиститись від циклічних атак. **Режим EAX** [4] EAX - режим автентифікованого шифрування з використанням нонсу (nonce). Тобто цей режим забезпечує конфіденційність(приватність), в сенсі неможливості відрізнити від випадокових бітів, та цілісність, в сенсі неможливості зловмисника(супротивника) виконати підробку повідомлення, тобто подати кортеж нової валідної з трійки $\langle N, H, C \rangle$, де N - нонс, H - асоційовані дані та C - шифртекст.

$$E : Key \times \{0,1\}^n \to \{0,1\}^n, \tau \in [0..n]$$

де E - алгоритм шифурвання, K - ключовий простір і τ - довжина тагу, тобто коду автентифікації. Ці параметри повинні бути зафіксовані перед відкриттям каналу комунікації. Позначимо таку схему $EAX\,[E, \tau]$ сигнатура цієї функції така ж ,як в (5) - (7). Робота схеми EAX представлена на Рис. 2.3. Базові блоки схеми Рис. 2.2

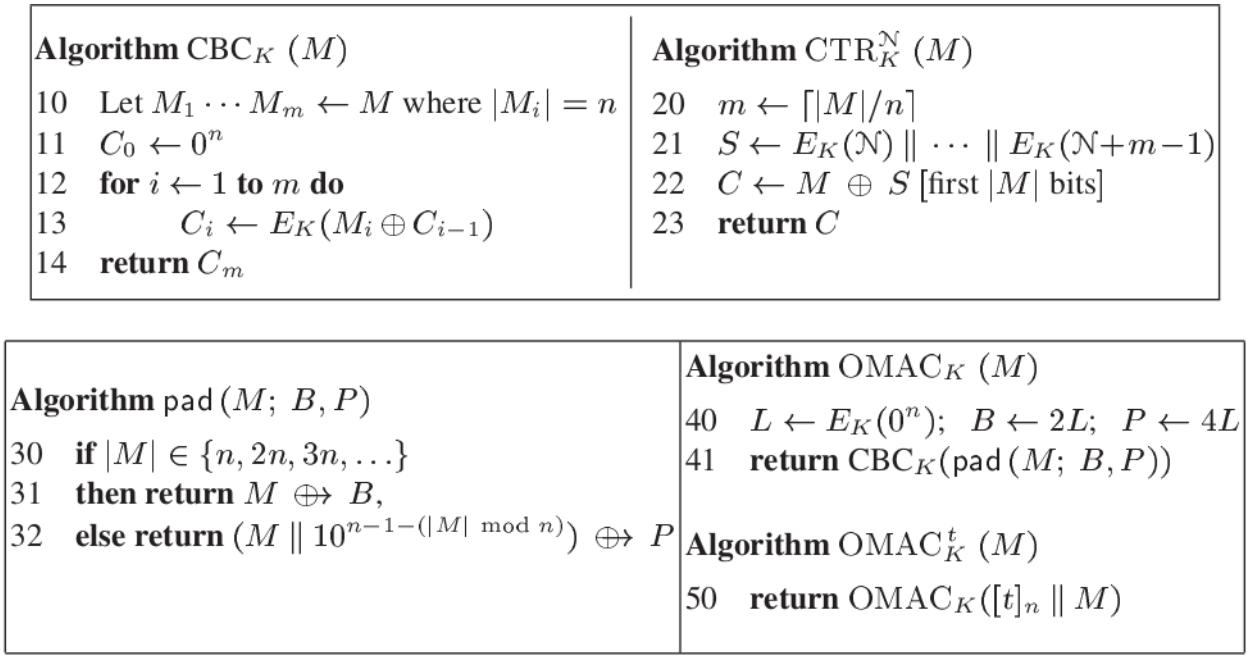


Рисунок 2.2 – Базові блоки схеми. E, $K \in Key$, для CBC $m \in (\{0,1\}^n)^+$. Для CTR $M \in \{0,1\}^*$ і $\mathbb{N} \in \{0,1\}^n$. Для $pad(\cdot)$ $M \in \{0,1\}^n$ і B, $P \in \{0,1\}^n$. Для $OMAC$ $M \in \{0,1\}^*$ і $t \in [0..2^n - 1]$ і множення на число L в полі $GF(2^n)$

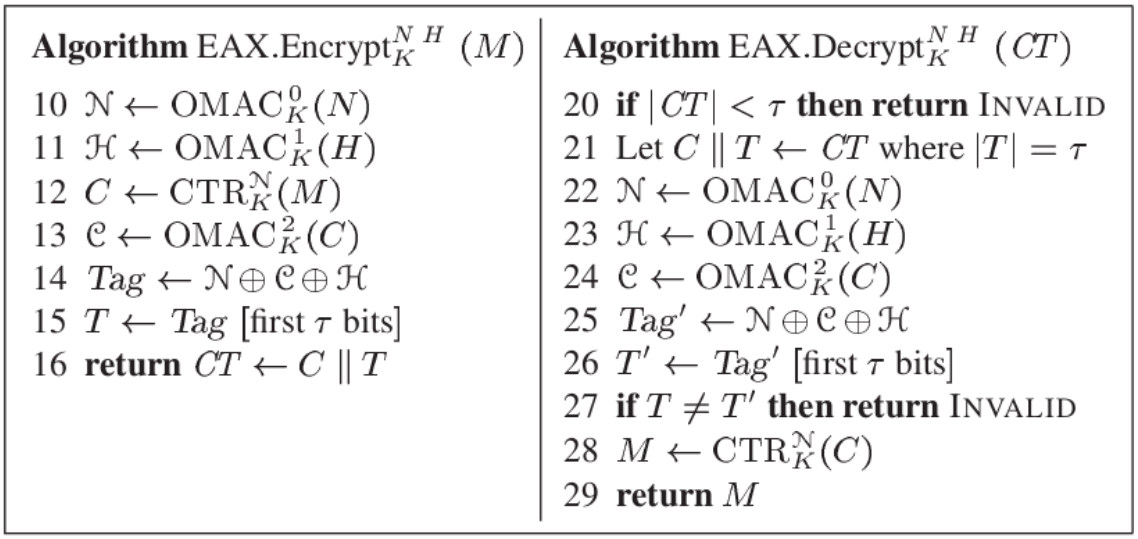


Рисунок 2.3 – Робота режиму EAX : шифрування і розшифрування. $M \in \mathcal{M}$ - повідомлення, $CT \in \mathcal{C}$ - шифротекст, N - нонс, H - асоційовані дані. E - алгоритм шифрування

Специфіка EAX

Чому автори не дотримались узагальненої схеми ?.

Насправді є обґрунтовані аргументи на користь загальна схема, заснованої на естетичних чи архітектурних сенсах. Можна стверджувати, що загальна схемакраще відокремлює концептуально незалежні елементи (конфіденційність та автентичність) і, відповідно, дозволяє підвищити гнучкість реалізації. Коректність також стає набагато простішою і зрозумілішою. Тим не менш, режими AEAD на основі блочних алгоритмів шифрування не мають деяких важливих переваг. Вони полегшують розробникам використання схеми, не знаючи багато криптографії, представляючи більш просту межу абстракції. Вони полегшують взаємодію. Вони знижують ризик того, що розробники виберуть небезпечні параметри. Вони можуть економити на ключових бітах та час настройки ключа, оскільки для методів загальна схеманезмінно потрібна пара окремих ключів.

На мою думку, автори EAX просто хотіли "втерти носа"схемі CCM, яка на той час мала успіх і використовувалась для IEEE 802.11 бездротових локальних мереж та 802.15.4 бездротових особистих мережевих зон.

Зокрема в наступному розділі розглядається EAX_2, як схема, що напряму наслідується від узагальненої схеми .

Потоковість EAX. Мається на увазі, що схема EAX може обробляти дані потоком, тобто обробка відбувається по надходженню даних і з використанням $\Theta(1)$ пам'яті, схема може і не знати, коли закінчиться потік даних. Зокрема, зауважимо, що потокові методи не повинні вимагати знання довжини повідомлення до тих пір, поки повідомлення не закінчиться. Невдача в потоковому режимі розцінюється як істотний дефект схеми шифрування або MAC. Більшість контекстів виконання (тобто забезпечення конфіденційності і автентичності повідомлення) заздалегідь відомо довжина рядка. Наприклад, у багатьох протоколах довжина рядка вже включається на нижньому рівні. Насправді така специфіка виконання представлена в обчислювальній

системі як послідовність байтів і кількість цих байтів. Але є й контексти, коли людина не знає заздалегідь довжину повідомлення, щоб отримати вказівку, що вона закінчена. Наприклад, рядки для друку часто представлені в комп'ютерних системах як послідовність ненульових байтів, а потім кінцевий нульовий байт. Безумовно, слід мати можливість ефективно зашифрувати рядок, представлений таким чином.

Обробка статичних даних

У багатьох сценаріях асоційовані дані A (автори EAX позначають H) будуть статичними протягом сеансу зв'язку. Наприклад, асоційовані дані можуть включати інформацію, таку як IP-адреса відправника, приймача та фіксовані криптографічні параметри - заголовки протоколу зв'язку -, пов'язані з цим сеансом. У такому випадку хочеться, щоб кількість часу для обчислення $E_K^{N\ A}(M)$ та $D_K^{N\ A}(C)$ не враховувала роботу над асоційованими даними. Що EAX і задовольняє.

Зокрема, EAX дає можливість відхилити невалідні повідомлення в середині виконнання розшифрування. Це одна з переваг підходу
$Encrypt \to MAC$ від підходу $MAC \to Encrypt$. Аби ефективно отримати MAC, як псевдовипадкову функцію(ПВФ), що лежить в основі EAX, $MAC_K(A) = E_K^{0^n\ A}(\epsilon)$, де ϵ - пусте повідомлення.

EAX_2

EAX це EAX_2 для частного випадку з CTR шифруванням та $OMAC$ для автентифікації, звужений до використання одного ключа для автентифікації та шифрування.

Нехай $\mathcal{F} : KEY_1 \times \{0,1\}^* \to \{0,1\}^n$ це псевдовипадкова функція(ПВФ), $n \geq 2$. $\prod = \langle \mathcal{E}, \mathcal{D} \rangle$ це IV - схема щифрування з ключовим простором Key_2 та $IV \in \{0,1\}^n$. Тобто $\mathcal{E} : KEY_2 \times \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^*$ та $\mathcal{D} : KEY_2 \times \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^*$ для кожного $K \in KEY_2$, $N \in \{0,1\}^n$, $M \in \{0,1\}^*$ виконано $C = \mathcal{E}_K^N(M) \leftrightarrow \mathcal{D}_K^N(C) = M$

Нехай $\tau \leq n$. Далі, зафіксуємо $\mathcal{F}, \prod, \tau$ і об'явимо $AEAD$ схему : $EAX_2 [\prod, \mathcal{F}, \tau] = (EAX_2.\mathcal{E}, EAX_2.\mathcal{D})$

Далі, позначимо $\mathcal{F}_K^t(M) = F_K([t]_n \| M)$, $KEY = KEY_1 \times KEY_2$.

Вся схема позначена на Рис. 2.4

Стійкість EAX_2

Спершу розглянемо стійкість EAX_2 і далі приведемо результат tweakableрозширення $OMAC$. Ці результати напряму наслідуються для EAX_1

Спершу, нехай $EAX_2[\prod, \mathcal{F}, \tau] : \mathcal{F} \equiv \mathcal{R}_n^n$ - множина всіх функцій $f : \{0,1\}^n \to \{0,1\}^n$, тобто F_{K_1} випадкова функція з $\{0,1\}^n \to \{0,1\}^n$

Лема 2.1. ***Стійкість*** EAX_2 ***з*** $F_{K_1} \in_R \mathcal{R}_n^n$. *Нехай* $\prod$ *це* $\mathbb{IVE}$ *схема, що* $IV \in \{0,1\}^n$, $\tau \in [0..n]$. *Тоді*

$$\mathbf{Adv}^{priv}_{EAX_2[\prod,\mathcal{R}_n^n,\tau]}(t,q,\sigma) \leq \mathbf{Adv}^{priv}_{\prod}(t^{'},q,\sigma)$$

де $t^{'} = t + \mathcal{O}(\sigma)$

Доведення.

Нехай KEY_2 є ключовим простором схеми $\mathcal{IVE}$ та $\prod = \langle E, D\rangle$. Нехай A - супротивник, що атакує конфіденційність схеми $AEAD$, $EAX_2[\prod, \mathcal{R}_n^n, \tau] = \langle E, D\rangle$. Припустимо, що A робить не більше q запитів до оракулу, використовує не більше σ даних : $S(space) \leq \sigma$ та $T(time) \leq t$. Будуємо супротивника P, на базі A, визначений на Рис. 2.5, для атаки на конфіденційність $\prod$. Щодо конструкції P, перше твердження полягає в тому, що присвоєння значення $f([0]^n \| N) \leftarrow \mathcal{N}$, зроблене у відповіді на запит про оракул A, є корректним, оскільки $f([0]^n \| N) \leftarrow \mathcal{N}$ не було попередньо визначене.

Це так, оскільки A приймає nonce.

Твердження 2.1.

$$\Pr\left[K_2 \xleftarrow{\$} KEY_2 : P^{\mathbb{E}^{\$}_{K_2}(\cdot)} = 1\right] = \Pr\left[f \xleftarrow{\$} \mathcal{R}_n^n; K_2 \xleftarrow{\$} KEY_2 : A^{\mathbf{E}^{\cdot,\cdot}_{f,K_2}(\cdot)} = 1\right]$$

Твердження 2.2.

$$\Pr\left[K_2 \xleftarrow{\$} KEY_2 : P^{\$(\cdot)} = 1\right] = \Pr\left[f \xleftarrow{\$} \mathcal{R}_n^n; K_2 \xleftarrow{\$} KEY_2 : A^{\$^{\cdot,\cdot}(\cdot)} = 1\right]$$

Algorithm EAX2.Encrypt$^{N\,H}_{K1,K2}$ (M)

10 $\mathcal{N} \leftarrow F^0_{K1}(N)$
11 $\mathcal{H} \leftarrow F^1_{K1}(H)$
12 $C \leftarrow \mathcal{E}^{\mathcal{N}}_{K2}(M)$
13 $\mathcal{C} \leftarrow F^2_{K1}(C)$
14 $Tag \leftarrow \mathcal{N} \oplus \mathcal{C} \oplus \mathcal{H}$
15 $T \leftarrow Tag$ [first τ bits]
16 **return** $CT \leftarrow C \parallel T$

Algorithm EAX2.Decrypt$^{N\,H}_{K1,K2}$ (CT)

20 **if** $|CT| < \tau$ **then return** INVALID
21 Let $C \parallel T \leftarrow CT$ where $|T| = \tau$
22 $\mathcal{N} \leftarrow F^0_{K1}(N)$
23 $\mathcal{H} \leftarrow F^1_{K1}(H)$
24 $\mathcal{C} \leftarrow F^2_{K1}(C)$
25 $Tag' \leftarrow \mathcal{N} \oplus \mathcal{C} \oplus \mathcal{H}$
26 $T' \leftarrow Tag'$ [first τ bits]
27 **if** $T \neq T'$ **then return** INVALID
28 $M \leftarrow \mathcal{D}^{\mathcal{N}}_{K2}(C)$
29 **return** M

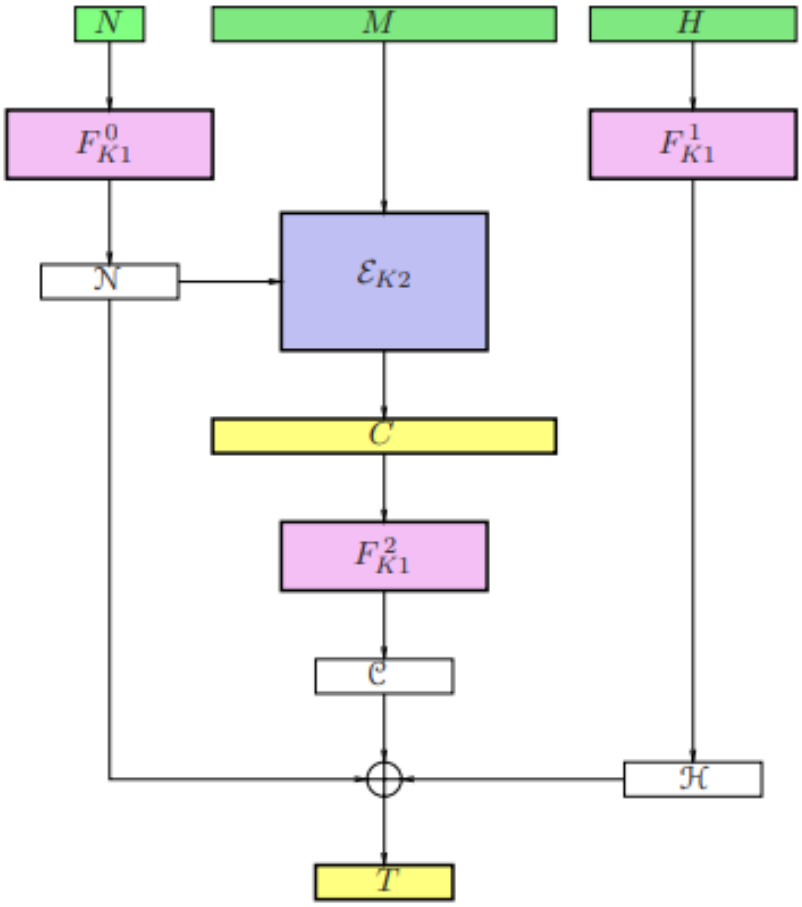


Рисунок 2.4 – Шифрування і розшифрування в схемі EAX_2

Перше твердження 2.1 зрозуміле з визначення. Щодо 2.2 повинні перевірити, що, коли супротивник P має оракул $\$(\cdot)$, то відповіді оракула рівномірно і незалежно розподілені, в формі $C\|T$. Ми знаємо, що шифротекст C випадковий вектор бітів. T таг також є випадковим, оскільки він - продукт виконання операцій xor деяких величин з $\mathcal{N}$ - що є випадковим. Віднімаємо і отримаємо результат.

$$\mathbf{Adv}^{priv}_{\prod}(P) = \mathbf{Adv}^{priv}_{EAX_2[\prod,\mathcal{R}^n_n,\tau]}(A)$$

Що і завершує твердження лемми.

□

Лема 2.2. *Нехай $m, \tau \geq 1$ цілі числа і A "rank-respecting"супротивник. Тоді*

$$\mathbf{Adv}^{xtag}_{m,\tau}(A) \leq 2^{-\tau}$$

Перед доведенням введемо.Нехай деяка гра, параметризована $m, \tau \geq 1$, $\mathcal{I}$ множина рядків довжини не більше m і $f : \mathcal{I} \to \{0,1\}^\tau$. Вважаємо, що супротивник посилає запит до двох оракулів $XTag^f(\cdot)$, $XVf^f(\cdot,\cdot)$. $XTag^f(\cdot)$ приймає на вхід множину $S \subseteq \mathcal{I}$ і повертає $\bigoplus_{x\in S} f(x)$.$XVf^f(\cdot,\cdot)$ приймає на вхід множину $S \subseteq \mathcal{I}$ і рядок T. Повертає 1, якщо $T = \bigoplus_{x\in S} f(x)$ і 0, якщо навпаки. Зокрема, коли пишемо суму бітових векторів, розуміємо як покомопонентне додавання за модулем 2. Також ставимо вимогу,що A надсилає тільки один запит до XVf і це останній запит до оракула. Тобто спочатку супротивник надсилає послідовність запитів до оракула $XTag$, потім до Xvf і припиняє свою роботу. Ми вважаємо, що A здійснив підробку, якщо XVf повертає 1.

$$\mathbf{Adv}^{xtag}_{m,\tau}(A) = \Pr\left[f \xleftarrow{\$} \mathcal{R}^{\mathcal{I}}_\tau : A^{XTag^f(\cdot),XVf^f(\cdot,\cdot)} forges\right]$$

Нехай, $c = |\mathcal{I}|$, $x_1, x_2, ..x_c$ - лексикографічний порядок над $\mathcal{I}$. Якщо

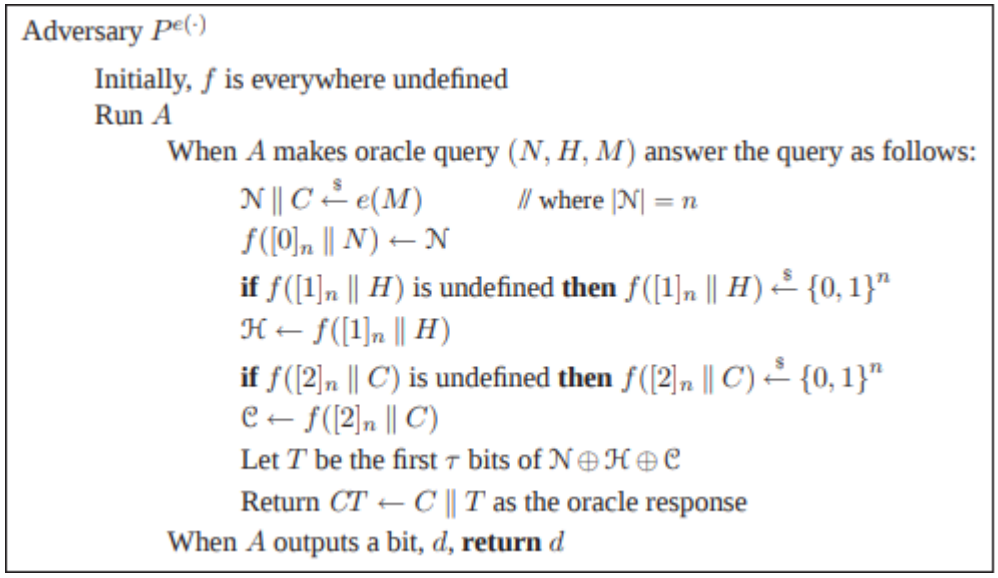


Рисунок 2.5 – Супротивник P, що атакує конфіденційність $\mathcal{IVE}$ схеми $\prod$, використовуючи A як оракул, що атакує конфіденційність $EAX_2\,[\prod, \mathcal{R}_n^n, \tau]$

$S \subseteq \mathcal{I}$, то позначимо $ChV(S)$ c-бітний характеристичний вектор, що означає $ChV(S)[j] = 1$, якщо $x_j \in S$ і 0, якщо навпаки (зрозуміло $1 \leq j \leq c$. І супротивник посилає запити $S_1, S_2, ..S_q$, (S, T) - XVf запит. Позначимо A, як *rank respecting*, якщо $ChV(S)$ це не лінійна комбінація з $ChV(S_1), ..ChV(S_q)$ - тобто лінійно незалежні. **Доведення.** Нехай S_i випадкова величина, що бере значення i-ого запиту $XTag$ оракулу $(1 \leq i \leq c-1)$, S_c - випадкова величина, що бере значення з першої компоненти пари запиту до XVf оракулу. Нехай $\mathbf{A_i}$ - випадкова величина, що бере значення з відповіді оракулу на S_i запит. Отож

$$A_i = \bigoplus_{x_i \in S_i} f(x) (1 \leq i \leq c)$$

Нехай A_c будь-який τ - бітовий рядок, A_i - відповідь оракула на i-ий запит.

$$\Pr\left[\mathbf{A_c} = A_c | (\mathbf{S_1}, ..\mathbf{S_c}, \mathbf{A_1}..\mathbf{A_{c-1}}) = (S_1, ..S_c, A_1, ..A_{c-1})\right] = 2^{-\tau} \qquad (2.1)$$

Adversary $B^{\mathrm{XTag}^f(\cdot),\mathrm{XVf}^f(\cdot,\cdot)}$
$i \leftarrow 0$;
Run A
When A makes an xor-tag query S
if $\mathrm{ChV}(S)$ is linearly dependent on $\mathrm{ChV}(S_1),\ldots,\mathrm{ChV}(S_i)$
then Let $L \subseteq \{1,\ldots,i\}$ be such that $\mathrm{ChV}(S) = \sum_{l\in L} \mathrm{ChV}(S_l)$; $A_i \leftarrow \sum_{l\in L} A_l$
else $i \leftarrow i+1$; $S_i \leftarrow S$; $A_i \leftarrow \mathrm{XTag}^f(S_i)$
Return A_i to A as the oracle response
When A makes an xor-verify query (S,T)
for $j = i+1,\ldots,c-1$ **do**
Pick some $S_j \subseteq I$ such that $\mathrm{ChV}(S),\mathrm{ChV}(S_1),\ldots,\mathrm{ChV}(S_j)$ are linearly independent
$A_j \leftarrow \mathrm{XTag}^f(S_j)$
Return $\mathrm{XVf}^f(S,T)$ to A as the oracle response

Рисунок 2.6 – Супротивник в доведенні лемми 2.2

$$\begin{aligned}
&\Pr\left[\mathbf{A_c} = A_c | (\mathbf{S_1},..\mathbf{S_c},\mathbf{A_1}..\mathbf{A_{c-1}}) = (S_1,..S_c,A_1,..A_{c-1})\right] \\
&= \frac{\left|\{f \in \mathcal{R}_\tau^{\mathcal{I}} : \bar{M}\cdot f = (A_1,..A_c)\}\right|}{\left|\{f \in \mathcal{R}_\tau^{\mathcal{I}} : M\cdot f = (A_1,..A_{c-1})\}\right|} \\
&= \frac{\left|\{f \in \mathcal{R}_\tau^{\mathcal{I}} : \bar{M}\cdot f = (A_1,..A_c)\}\right|}{\sum_{A\in\{0,1\}^\tau}\left|\{f \in \mathcal{R}_\tau^{\mathcal{I}} : \bar{M}\cdot f = (A_1,..A_{c-1},A)\}\right|} \\
&= \frac{1}{\sum_{A\in\{0,1\}^\tau 1}} = \frac{1}{2^\tau}
\end{aligned}$$

□

Лема 2.3. *Нехай* $\prod$ *це* $\mathcal{IVE}$ *схема,* $IV \in \{0,1\}^n$, $\tau \in [0..n]$. *Тоді*

$$\mathbf{Adv}^{priv}_{EAX_2[\prod,\mathcal{R}^n_n,\tau]}(t,q,\sigma) \le \mathbf{Adv}^{priv}_{\prod}(t^{'},q,\sigma) + 2^{-\tau}$$

де $t^{'} = t + \mathcal{O}(\sigma)$

Доведення. Нехай,B супротивник, що намагається виконати підробку схеми $EAX_2\left[\prod,\mathcal{R}^n_n,\tau\right]$. Припустимо, він виконує не більше q_e запитів до шифруючого оракула, складність по памяті $S \le \sigma$, $T(time) \le t$. m достатньо велике число, що жоден запит до оракулу не має довжину більше за m. $\mathcal{I}$ - множина усіх рядків довжини, що не

більше $m.\forall f : \mathcal{I} \to \{0,1\}^n$: на Рис. 2.7

$$\mathbf{Adv}^{rauth}(B) = \Pr\left[f \xleftarrow{\$} \mathcal{R}_n^{\mathcal{I}} : B^{\mathbf{EE_f}(\cdot),\hat{DD}_f(\cdot)} \text{ виконав підробку}\right]$$

Далі, конструюємо "rank-respecting"супротивника, що

$$\mathbf{Adv}^{rauth}(B) \leq \mathbf{Adv}^{xtag}_{m,\tau}(A) \tag{2.2}$$

Зокрема, сконструюємо супротивника P, що використовує ресурси (t', q, σ) і намагається виконати підробку :

$$\mathbf{Adv}^{auth}_{EAX_2[\prod,\mathcal{R}_n^n,\tau]}(B) - \mathbf{Adv}^{rauth}(B) \leq \mathbf{Adv}^{priv}_{\prod}(P) \tag{2.3}$$

Маємо,

$$\mathbf{Adv}^{auth}_{EAX_2[\prod,\mathcal{R}_n^n,\tau]} =$$

$$\mathbf{Adv}^{rauth}(B) + \left(\mathbf{Adv}^{auth}_{EAX_2[\prod,\mathcal{R}_n^n,\tau]}(B) - \mathbf{Adv}^{rauth}(B)\right)$$

$$\leq \mathbf{Adv}^{xtag}_{m,\tau}(A) + \mathbf{Adv}^{priv}_{\prod}(P) \ \leq 2^{-\tau} + \mathbf{Adv}^{priv}_{\prod}(t', q, \sigma)$$

Супротивник $A^{Xtag^f(\cdot),XVf^f(\cdot,\cdot)}$ визначає такі підпрограми, оракули на Рис. 2.8 :

Супротивник A потім викликає $B^{SimE(\cdot),SimD(\cdot)}$. Тоді рівняння 2.2 коректне, оскільки для будь-якого f маємо : $SimE(\cdot) = \mathbf{EE}_f(\cdot)$

Algorithm $\mathbf{EE}_f^{N\,H}(M)$

$\mathcal{N} \leftarrow f([0]_n \parallel N)$
$\mathcal{H} \leftarrow f([1]_n \parallel H)$
$C \xleftarrow{\$} \{0,1\}^{|M|}$
$\mathcal{C} \leftarrow f([2]_n \parallel C)$
$Tag \leftarrow \mathcal{N} \oplus \mathcal{C} \oplus \mathcal{H}$
$T \leftarrow Tag$ [first τ bits]
return $CT \leftarrow C \parallel T$

Algorithm $\widehat{\mathbf{DD}}_f^{N\,H}(CT)$

if $|CT| < \tau$ **then return** INVALID
Let $C \parallel T \leftarrow CT$ where $|T| = \tau$
$\mathcal{N} \leftarrow f([0]_n \parallel N)$
$\mathcal{H} \leftarrow f([1]_n \parallel H)$
$\mathcal{C} \leftarrow f([2]_n \parallel C)$
$Tag' \leftarrow \mathcal{N} \oplus \mathcal{C} \oplus \mathcal{H}$
$T' \leftarrow Tag'$ [first τ bits]
if $T \neq T'$ **then return** INVALID **else return** 1

Рисунок 2.7

Subroutine $\mathrm{SimE}^{N\,H}\ (M)$

$C \xleftarrow{\$} \{0,1\}^{|M|}$
$S \leftarrow \{\,[0]_n \,\|\, N,\ [1]_n \,\|\, H,\ [2]_n \,\|\, C\,\}$
$T \leftarrow \mathrm{XTag}^f(S)$
return $CT \leftarrow C \,\|\, T$

Subroutine $\mathrm{SimD}^{N\,H}\ (CT)$

if $|CT| < \tau$ **then return** INVALID
Let $C \,\|\, T \leftarrow CT$ where $|T| = \tau$
$S \leftarrow \{\,[0]_n \,\|\, N,\ [1]_n \,\|\, H,\ [2]_n \,\|\, C\,\}$
if $\mathrm{XVf}^f(S,T) = 0$ **then return** INVALID
else return 1

Рисунок 2.8

$SimD(\cdot) = \hat{DD}_f(\cdot)$. Також, A є "rank-respecting". Нехай, $1 \leq i \leq q$ (N_i, H_i, M_i) буде i-тим запитом B до оракулу-шифрування, виданий директивою A для XVf оракулу (S,T). $CT = C\|T$, де $|T| = \tau$. Уявіть матрицю, i-й ряд якої - $ChV(S_i)$ $(1 \leq i \leq q$, а чий $(q+1)$ -й рядок - $ChV(S)$. Стовпчик j називається l-стовпцем, якщо x_j з префіксом $[l]_n$ $(0 \leq l \leq 2$ та $1 \leq j \leq c)$. Оскільки A є nonce схемою, то існує множина D з q 0-стовпців, така що підматриця, утворена першими q рядками матриці, і стовпці в D є матрицею одиниць $q \times q$. Оскільки $ChV(S)$ має рівно один 1 у 0-стовпчику, єдиний спосіб, коли $ChV(S)$ може бути лінійною комбінацією $ChV(S_1), ..., ChV(S_q)$ - це те, що він дорівнює $ChV(S_i)$ для деяких i : $(1 \leq i \leq q)$. Це означає, що $N = N_i$, $H = H_i$, і відповідь на i-й запит оракула B був CT. Але це суперечить умові, яку ми наклали на B, забороняючи запит перевіряючого оракулу (N, H, CT), таким чином, що CT був отриманий у відповідь на запит шифрувального оракулу (N, H, M). (Тут важливо, щоб ми вимагали виконання умови, незалежно від відповідей на запити оракул та підкидання монети) Отже, $ChV(S)$ не може дорівнювати $ChV(S_i$. Це завершує доказ того, що A "rank-respecting".

Щодо конструкції супротивника P. Це зображено на малюнку 2.9. Найважливішою особливістю EAX_2, яку ми використали для того, щоб мати можливість відповідати на запити Xvf, є те, що валідність шифротексту можна перевірити, не розшифровуючи за схемою $\mathcal{IVE}$. Що стосується конструкції P, перше твердження полягає в тому, що її

присвоєння значення $f([0]_n, N)$, зроблене у відповіді на запит оракулу шифрування B, є коректним, оскільки $f([0]_n, N)$ раніше не було визначено. Це вірно з двох причин. Перший полягає в тому, що B nonce-схема. Друга полягає в тому, що B не робить жодних запитів до оракулу шифрування після того, як він зробив свій запит до XVf. (Запит XVf може визначати $f([0]_n, N)$, але оскільки жодних запитів до оракулу шифрування не слідує, ми не повинні турбуватися про те, що $f([0]_n, N)$ визначається під час відповіді на один з них). Далі, нехай KEY_2 ключовий простір $\prod$. Тоді

$$\Pr\left[K_2 \xleftarrow{\$} KEY_2 : P^{\mathcal{E}^{\$}_{K_2}(\cdot)} = 1\right] = \mathbf{Adv}^{auth}_{EAX_2[\prod,\mathcal{R}^n_n,\tau]}(B)$$

$$\Pr\left[K_2 \xleftarrow{\$} KEY_2 : P^{\$(\cdot)} = 1\right] = \mathbf{Adv}^{rauth}_{EAX_2[\prod,\mathcal{R}^n_n,\tau]}(B)$$

□

Теорема 2.6. *Нехай* $F : KEY_1 \times \{0,1\}^* \to \{0,1\}^n$ *сім'я функцій,* $\prod = \langle \mathcal{E}, \mathcal{D} \rangle$ *це* $\mathcal{IVE}$ *схема з IV простором* $\{0,1\}^n$ *і* $\tau \in [0..n]$. *Тоді*

$$\mathbf{Adv}^{priv}_{EAX_2[\prod,\mathcal{R}^n_n,\tau]}(t,q,\sigma)$$

$$\leq \mathbf{Adv}^{priv}_{\prod}(t_2,q,\sigma) + \mathbf{Adv}^{ПВФ}_{F}(t_1,3(q+1),\sigma) + 2^{-\tau}$$

$$\mathbf{Adv}^{priv}_{EAX_2[\prod,\mathcal{R}^n_n,\tau]}(t,q,\sigma) \leq \mathbf{Adv}^{priv}_{\prod}(t_2,q,\sigma) + \mathbf{Adv}^{ПВФ}_{F}(t_3,3q,\sigma)$$

, де $t_1 = t + Time_\epsilon(\sigma) + \mathcal{O}(\sigma)$, $t_2 = t + \mathcal{O}(\sigma + n * q)$, $t_3 = t + Time_\epsilon(\sigma) + \mathcal{O}(\sigma)$

Доведення. Нехай A супротивник використовує ресурси (t,q,σ) і намагається виконати підробку $\prod = \langle \mathbf{E}, \mathbf{D} \rangle = EAX_2[\prod, F, \tau]$. Використовуючи A можемо виконати побудову супротивника для того, щоб розрізняти $f \xleftarrow{\$} F$ від $f \xleftarrow{\$} \mathcal{R}^n_n$. На початку роботи B робить вибір $K_2 \xleftarrow{\$} KEY_2$ - ключовий простір $\prod$. Потім B запускає A. Коли A робить запит до оракулу (N_i, H_i, M_i) суперник обчислює $\mathcal{N}_i \leftarrow f([0]_n \| N_i)$, $C_i \leftarrow \mathcal{E}^{\mathcal{N}_i}_{K_2}(M_i)$ і $\mathcal{H}_i \leftarrow f([1]_n \| H_i), \mathcal{C}_i \leftarrow f([2]_n \| C_i), T_i \leftarrow (\mathcal{N}_i \oplus \mathcal{C}_i \oplus \mathcal{H}_i[\text{перші } \tau \text{ бітів}])$,

Adversary $P^{e(\cdot)}$

Initially, f is everywhere undefined
Run B

When B makes encryption-oracle query (N, H, M):

$\mathcal{N} \parallel C \xleftarrow{\$} e(M)$ // where $|\mathcal{N}| = n$
$f([0]_n \parallel N) \leftarrow \mathcal{N}$
if $f([1]_n \parallel H)$ is undefined **then** $\mathcal{H} \leftarrow f([1]_n \parallel H) \xleftarrow{\$} \{0,1\}^n$
if $f([2]_n \parallel C)$ is undefined **then** $\mathcal{C} \leftarrow f([2]_n \parallel C) \xleftarrow{\$} \{0,1\}^n$
Let T be the first τ bits of $\mathcal{N} \oplus \mathcal{H} \oplus \mathcal{C}$
Return $CT \leftarrow C \parallel T$ to B as the oracle response

When B makes verification-oracle query (N, H, CT):

if $|CT| < \tau$ **then return** INVALID to B as the oracle response
Let $C \parallel T \leftarrow CT$ where $|T| = \tau$
if $f([0]_n \parallel N)$ is undefined **then** $f([0]_n \parallel N) \xleftarrow{\$} \{0,1\}^n$
$\mathcal{N} \leftarrow f([0]_n \parallel N)$
if $f([1]_n \parallel H)$ is undefined **then** $f([1]_n \parallel H) \xleftarrow{\$} \{0,1\}^n$
$\mathcal{H} \leftarrow f([1]_n \parallel H)$
if $f([2]_n \parallel C)$ is undefined **then** $f([2]_n \parallel C) \xleftarrow{\$} \{0,1\}^n$
$\mathcal{C} \leftarrow f([2]_n \parallel C)$
Let T' be the first τ bits of $\mathcal{N} \oplus \mathcal{H} \oplus \mathcal{C}$
if $T = T'$ **then** $d \leftarrow 1$ **else** $d \leftarrow 0$
if $d = 0$
then return INVALID to B as the oracle response
else return 1 to B as the oracle response

return d

Рисунок 2.9

$CT_i \leftarrow C_i \| T_i$ і тоді B повертає до A рядок CT_i. Коли A завершує свою роботу і видає спробу підробки $(N, H, C\|T)$, то B перевіряє, чи це валідга підробка: він перевіряє, чи (N, H, CT) відрізняється від кожного $(N_i, H_i, C_i\|T_i)$, який був обчислений; він обчислює $\mathcal{N} \leftarrow f([0]_n\|N)$ і $\mathcal{H} \leftarrow f([1]_n\|H)$ і $\mathcal{C} \leftarrow f([2]_n\|C)$ і перевіряє, чи $T = (N \oplus H \oplus)$[перші τ бітів]. Якщо обидві умови виконані, то B повертає біт 1 (він здогадується, що $f = F_{K_1}$ для випадкового K_1), інакше він виводить біт 0 (він здогадується, що $f \xleftarrow{\$} \mathcal{R}_n^n$). Зауважимо, що B здійснює в загальному $3q + 3$ викликів оракула. Загальна довжина цих запитів - σ. (Згадаймо нашу умову, що ми включаємо в σ вихідну довжину та кількість компонентів у кожному векторі, який запитується.) Час роботи B - $t_1 = t + Time_\epsilon(\sigma) + \mathcal{O}(\sigma)$. Нарешті, супротивник B надає A чудову модель $EAX_2[\prod, F, \tau]$, якщо $f \xleftarrow{\$} F$, поки B надає A модель $EAX_2[\prod, \mathcal{R}_n^n, \tau]$, якщо $f \xleftarrow{\$} \mathcal{R}_n^n$. Таким чином, використовуючи лему 2.3 Маємо :

$$\mathbf{Adv}^{auth}_{EAX_2[\prod,F,\tau]}(A)$$

$$= (\mathbf{Adv}^{auth}_{EAX_2[\prod,F,\tau]}(A) - \mathbf{Adv}^{auth}_{EAX_2[\prod,\mathcal{R}_n^n,\tau]}(A)) + \mathbf{Adv}^{auth}_{EAX_2[\prod,\mathcal{R}_n^n,\tau]}(A)$$

$$\leq ([f \xleftarrow{\$} FLB^f = 1] - [f \xleftarrow{\$} \mathcal{R}_n^n : B^f = 1]) + 2^{-\tau} + \mathbf{Adv}^{priv}_{\prod}(t_2, q, \sigma)$$

$$= \mathbf{Adv}^{\text{ПВФ}}_{F}(B) + \mathbf{Adv}^{priv}_{\prod}(t_2, q, \sigma) + 2^{-\tau}$$

$$\leq \mathbf{Adv}^{\text{ПВФ}}_{F}(t_1, 3(q+1), \sigma) + \mathbf{Adv}^{priv}_{\prod}(t_2, q, \sigma) + 2^{-\tau}$$

Для доведення другої рівності леми, використовуємо ту ж саму конструкцію з деякими змінами. Тепер, коли A завершує роботу, виводить біт b, противник B видає той самий біт b. Загальна кількість запитів до оракула, виконаних B, становить $3q$. Загальна довжина цих запитів не більше σ. Час роботи B - $t_3 = t + Time_\epsilon(\sigma) + \mathcal{O}(\sigma)$. Тепер використовуємо лему 2.1

$$\mathbf{Adv}^{priv}_{EAX_2[\prod,F,\tau]}(A)$$

$$= (\mathbf{Adv}^{priv}_{EAX_2[\prod,F,\tau]}(A) - \mathbf{Adv}^{priv}_{EAX_2[\prod,\mathcal{R}^n_n,\tau]}(A)) + \mathbf{Adv}^{priv}_{EAX_2[\prod,\mathcal{R}^n_n,\tau]}(A)$$

$$\leq (\Pr[f \xleftarrow{\$} F : B^f = 1] - [f \xleftarrow{\$} \mathcal{R}^n_n : B^f = 1]) + 2^{-\tau} + \mathbf{Adv}^{priv}_{\prod}(t_2, q, \sigma)$$

$$= \mathbf{Adv}^{\text{ПВФ}}_{\prod}(B) + \mathbf{Adv}^{priv}_{\prod}(t_2, q, \sigma) \leq \mathbf{Adv}^{\text{ПВФ}}_{\prod}(t_3, 3q, \sigma) + \mathbf{Adv}^{priv}_{\prod}(t_2, q, \sigma)$$

□

Лема 2.4. ***Псевдовипадковість OMAC***. *Нехай* $n \geq 2$ *Тоді для супротивників обмеженим розміром запитів*

$$\mathbf{Adv}^{dist}_{\mathbb{OMAC}[\mathcal{R}^n_n],\$_n}(\sigma_1, \sigma_2) \leq (\sigma_1 + \sigma_2 + 3)^2 \cdot 2^{-n}$$

Теорема 2.7. ***Стійкість*** EAX. *Нехай,* $n \geq 2, \tau \in [0..n]$ *Тоді*

$$\mathbf{Adv}^{priv}_{EAX[\mathcal{R}^n_n,\tau]}(\sigma) \leq \frac{9\sigma^2}{2^n}$$

$$\mathbf{Adv}^{auth}_{EAX[\mathcal{R}^n_n,\tau]}(\sigma) \leq \frac{10.5\sigma^2}{2^n} + 2^{-\tau}$$

Доведення. Нехай A - супротивник, що використовує ресурси (q, σ), який намагається відрізнити $EAX[\mathcal{R}^n_n, \tau]$ від джерела випадкових бітів. Побудуємо супротивника B, який відрізняє $\mathbb{OMAC}[\mathcal{R}^{n_n}]$ від джерела випадкових бітів. Супротивник має оракул g, який відповідає на запити $(t, M, s) \in \{0, 1, 2\} \times \{0, 1\}^* \times N$ з рядком $RS_0S_1...S_{s1}$, кожен з яких є n-бітним рядком. Супротивник працює наступним чином Рис 2.10 : Нехай супротивник A робить $q \geq 1$ запитів. За домовленістю супротивник B використовує ресурсів не більше $(2\sigma - 3, \sigma)$. Зауважте, що $\Pr[A^{EAX[\mathcal{R}^n_n,\tau]} = 1] = \Pr[B^{\mathbb{OMAC}[R^n_n]} = 1]$. Крім того, оскільки A "nonce"схема, B - з обмеженими за довжиною запитами, то $\Pr[A^{\$} = 1] = \Pr[B^{\$_n} = 1]$. Використовуючи лему 2.4

$$\mathbf{Adv}^{priv}_{EAX[\mathcal{R}^n_n,\tau]}(A) = \Pr[A^{EAX[\mathcal{R}^n_n,\tau]} = 1] - \Pr[A^{\$} = 1]$$

$$= \Pr[B^{\mathbb{OMAC}[\mathcal{R}^n_n]} = 1] - \Pr[B^{\$_n} = 1]$$

Algorithm B^g

Run A

When A makes an oracle call $(N_i,\ H_i,\ M_i)$, do the following:

$s \leftarrow \lceil |M_i|/n \rceil$

$\mathcal{N}_i\, S_0 \ldots S_{s-1} \leftarrow g(0,\ N_i,\ s)$

$C_i \leftarrow M_i \oplus (S_0 \cdots S_{s-1}$ [first $|M_i|$ bits])

$\mathcal{H}_i \leftarrow g(1,\ H_i,\ 0)$

$\mathcal{C}_i \leftarrow g(2,\ C_i,\ 0)$

$T_i \leftarrow \mathcal{N}_i \oplus \mathcal{C}_i \oplus \mathcal{H}_i$ [first τ bits]

Return, in response to A's query, $C_i \,\|\, T_i$

When A halts, outputting a bit b, **return** b

Рисунок 2.10

$$\leq\ \mathbf{Adv}^{dist}_{\mathbb{OMAC}[\mathcal{R}^n_n,\$_n]}(2\sigma-3,\sigma)$$

$$\leq\ (3\sigma)^2\cdot 2^{-n}\ \leq\ 9\sigma^2\cdot 2^{-n}$$

Тепер щодо автентифікації, нехай A супротивник, що намагається виконати підробку, використовує не більше (σ) ресурсів. Нехай,

$$\alpha_1 = \mathbf{Adv}^{auth}_{EAX[\mathcal{R}^n_n,\tau]}(A)\alpha_2 = \mathbf{Adv}^{auth}_{EAX_2[CTR[\mathcal{R}^n_n],\mathcal{R}^*_n,\tau]}(A)$$

З леми 2.3 і відомих результатів :

$$\alpha_2 \leq 2^{-\tau} + \mathbf{Adv}^{priv}_{CTR[\mathcal{R}^n_n]}(\sigma) \leq 2^{-\tau} + \sigma^2\cdot 2^{-n}$$

Маємо,

$$\alpha_1 = \alpha_2 + \delta \leq \delta + \sigma^2\cdot 2^{-n} + 2^{-\tau}$$

Визначимо таку функцію $E\,[\rho, f] = \{0,1,2\}\times\{0,1\}^*\times\mathbb{N}\to\{0,1\}^*$ Тепер

$$\alpha_2 = \mathbf{Adv}^{auth}_{EAX_2[CTR[\mathcal{R}^n_n],\mathcal{R}^*_n,\tau]}(A) = \Pr\left[B^{E[\mathcal{R}^n_n,\mathcal{R}^*_n]} = 1\right]$$

Зокрема, $\mathbf{Adv}^{dist}_{E[\mathcal{R}^n_n,\mathcal{R}^*_n],\$_n} \leq \sigma_2^2\cdot 2^{-n-1}$. Будь-яких супротивників, які роблять загалом σ_2 запитів, оскільки $E[\mathcal{R}^n_n, R_n]$ можна відрізнити від $\$_n$ лише за

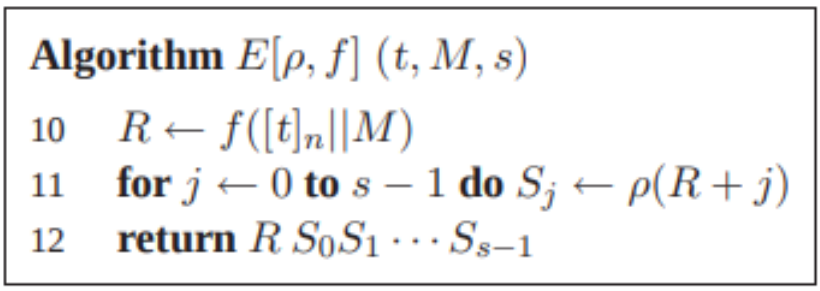


Рисунок 2.11

умови колізії входів до ρ і σ_2 входів до ρ. Як наслідок,

$$\Pr\left[B^{E[\mathcal{R}_n^n, R_n]} = 1\right] \;\geq\; \Pr\left[B^{\$n} = 1\right] - \sigma^2 \cdot 2^{-n-1}$$

Звідси

$$\alpha_2 = \mathbf{Adv}^{auth}_{EAX_2[CTR[\mathcal{R}_n^n], \mathcal{R}_n^*, \tau]}(A) \;\geq\; \Pr\left[B^{\$n} = 1\right] - \sigma^2 \cdot 2^{-n-1}$$

Використовуємо той факт, що останні три запити B мають форму $g(\cdot, \cdot, 0)$, тож останні три запити не можуть порушити умови щодо обмеження довжини запитів.Тож, використовуючи лему 2.4,

$$\delta = \alpha_1 - \alpha_2 \leq 9 \cdot \sigma^2 \cdot 2^{-n} + \sigma^2 \cdot 2^{-n-1} \leq 9.5\sigma^2 \cdot 2^{-n}$$

□

Висновки до розділу 2

В цьому розділі були представлені підходи до побудови алгоритмів автентифікованого шифрування, а також теоретичні напрацювання в цій області дослідження. Тобто були приведені аргументи щодо використання узагальненої схеми побудови $AEAD$ це $Encrypt - then - Mac(EtM)$, шла мова про більш сильні властивості, наприклад $CUF - CPA$, стійкість до підробки шифротексту, тобто якщо $Mac - then - Encrypt$ схема має таку властивість, то має гарну криптографічну стійкість і може

використовуватись для побудови захищених каналів зв'язку. Зокрема було надано детальний огляд стійкості і швидкодії таких режимів $AEAD$, як GCM, EAX. Приведено підхід до побудови схеми $AEAD$ з суматорами і теорема про стійкість останніх.

3 ОСНОВНІ РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Підсумуємо відомості щодо сучасних режимів автентифікованого шифрування.

– GCM

Galois Counter Mode став найпопулярнішим режимом $AE(AD)$ на сьогодні. Популярність частково пояснюється тим, що GCM надзвичайно швидкий, але здебільшого це відбувається тому, що режим незапантенований. GCM може працювати в режимі обробки потоку даних ("онлайн"), тобто вам не потрібно наперед знати довжину повідомлення, може бути паралелізований, останні версії $OpenSSL$ і $Crypto++$ забезпечують відмінну реалізацію режима, головним чином тому, що він зараз підтримується як TLS. Одним з суттєваих недоліків GCM - складність реалізації, оскільки це CTR режим шифрування з додаванням коду автентифікації типу Carter-Wegman [18], [17], [5]

– EAX

EAX - two-pass схема, що означає, що шифрування та аутентифікація здійснюються в окремо, послідовно. Це робить його повільніше, ніж GCM або OCB, хоча (на відміну від CCM) це "онлайн"схема. Проте EAX має переваги: незапентований, досить просто реалізувати, використовуючи лише блочний алгоритм шифрування, це означає, що ви можете технічно вписати його в реалізацію з дуже обмеженим розміром коду.

– CCM

CCM, як і EAX, також two-pass схема. Однак, на відміну від останнього, має декілька суттєвих недоліків. Перший і, на мою думку, найбільш суттєвий це необхідність компромісу між максимальной довжиною повідомлення і довжиною nonce схеми. Ці два параметри взагалі не повинні бути пов'язані. По-друге, схема CCM не є "онлайн тобто перед початком роботи користувачу необхідно знати довжину повідомлення. Зокрема, не підпримується наявність передобробки статичних даних,

тобто нам би хотілось, аби на початку комунікації з іншим користувачем - на початку сесії - обробка статичних даних таких, як заголовки протоколів прикладного рівня, провелась тільки один раз і не впливала на майбутні обчислення. По суті, причиною того, що CCM не дозволяє попередньо обробити статичні асоційовні дані H, полягає в тому, що алгоритм кодує значення nonce і довжину повідомлення $\|M\|_n$ перед H, а не після нього.[4]

3.1 Суматори AEAD

Ідея криптографічних суматорів полягає у внесені надмірності у безпеку шляхом об'єднання декількох конструкцій одного і того ж примітиву в один, у випадку, коли один або декілька цих будівельних блоків виявляться нестійкими. Поки щонайменше один компонент залишається криптографічно стійким, залишається стійкою і побудована конструкція. Суматори були вивчені для численних криптографічних примітивів, включаючи односторонні функції, хеш-функції, коди аутентифікації повідомлень, підписи, симетричні схеми шифрування і багато інших [12]. Деякі з цих робіт більше з теоретичної сторони, а інші - пропонують рішення проблем на практиці.

Мотивація використання $AEAD$ суматорів

Дослідження симетричної криптографії вдалося запропонувати безліч схем AEAD. Незважаючи на те, що багато хто з них мають завірені проблеми з безпекою, все ж є ряд причин, чому поєднання двох схем $AEAD$ в одну для посилення їх безпеки може бути привабливим. Причини можна знайти в різних сферах, вони пов'язані з власне стійкістю, проблемами впровадження, практикою управління або просто політикою.

Порушення стійкості схеми. Навіть якщо криптографічна схема доказово стійка, прикрою правдою є те, що аргументи щодо її стійкості

можуть бути порушені, а схема виявитись слабкою. Це також стосується конструкцій $AEAD$, де, як видається, справедливо сказати, що високий рівень інтеграції, який характеризується найефективнішими схемами на основі алгоритмів блочного шифрування, також робить схеми складними для повного розуміння та оточення на формальному рівні. Прекрасний приклад того, коли стійкість схеми $AEAD$ було порушено, а саме $OCB2$ [8], робота, що повністю порушила схему.

Атаки на реалізацію. Навіть якщо конструкція поставляється з коректним формальним доказом стійкості, надійна її реалізація може бути складною. Наприклад, якщо компоненти примітиву зручно реалізовувати за допомогою таблиць, тоді важко уникнути атак по часу.

Стандарти. Для досягнення відповідності стандартам або для отримання сертифікації органами влади може знадобитися використання певної схеми $AEAD$. Це може бути так, навіть якщо ця схема $AEAD$ є слабкою або, принаймні, її безпека сумнівна. Приклади можуть включати випадки в галузі платежів, де використання $3DES$ все ще здається популярним, навіть якщо його розмір блоку є занадто малим для досягнення бажаною захищенності в багатьох застосувань, або де необхідне використання стандартизованих ISO-схем $AEAD$, таких як зламаний $OCB2$ і таке інше. Суматори $AEAD$ дозволяють використовувати слабкі, але необхідні схеми, без загроз безпеці.

Підходи до побудови $AEAD$ суматорів.

– Підхід чорної коробки(Blackbox)

В роботі [12] представлено суматор $AEAD$, який є повністю 'чорною коробкою', це означає, що він працює узагальнено для будь-яких двох $AEAD$ схем. Зрозуміло, що будь-який такий суматор щонайменше один раз посилається на кожну схему компонентів: комбіноване шифрування викликає алгоритм шифрування кожного компонента, таким ж чином працює і комбіноване розшифрування - виклик алгоритму дешифрування кожної компоненти. На рис. 3.1 описано схему підходу.

– На основі схеми $Encrypt-then-Mac$ з суматором 'чорна коробка'

Цей підхід переплітає EtM схему $AEAD$ AE_0, реалізовану алгоритмами $\langle E_0, D_0, M_0\rangle$ із схемою AEAD чорної коробоки $AE_1 = (enc_1, dec_1)$. Шифротекст повідомлення, яке ми шифруємо, за схемою EtM $c_0 \leftarrow E_0(k_e, n, m)$, потім подається в алгоритм шифрування 'чорної коробки'. До отриманого шифротексту застосовуємо функцію MAC разом із статичними даними.

Algorithm 3.1 Алгоритм конструкції EtM підходу для побудови $AEAD$ суматорів

Proc $enc(k, n, ad, m)$
$((k_e, k_m), k_1) \leftarrow k$
$c_0 \leftarrow E_0(k_e, n, m)$
$c_1 \leftarrow enc_1(k_1, n, ac, c_0)$
$t \leftarrow M_0(k_m, n\|ad\|c_1)$
$c \leftarrow c_1\|t$
Return c

Теорема 3.1. *Для кожної пари коректних $AEAD$ схем $\langle AE_0, AE_1\rangle$, для кожного $i \in \{0,1\}$ і кожного супротивника $\mathcal{A}$ на комбіновану схему $CB(AE_0, AE_1)$, як вказано в 3.1, існує супротивник $\mathcal{B}_i^{int}$, що $\mathbf{Adv}^{int}_{CB(AE_0,AE_1)}(\mathcal{A}) \leq \mathbf{Adv}^{int}_{AE_i}(\mathcal{B}_i^{int})$, існує супротивник $\mathcal{B}_0^{ind}$, що $\mathbf{Adv}^{ind}_{CB(AE_0,AE_1)}(\mathcal{A}) \leq \mathbf{Adv}^{ind}_{AE_0}(\mathcal{B}_0^{in})$ і існує супротивник $\mathcal{B}_1^{ind}$, що $\mathbf{Adv}^{ind}_{CB(AE_0,AE_1)}(\mathcal{A}) \leq \mathbf{Adv}^{int}_{AE_1}(\mathcal{B}_1^{int}) + \mathbf{Adv}^{ind}_{AE_1}(\mathcal{B}_1^{ind})$. Час виконання $\mathcal{B}$ супротивників той же, що і у $\mathcal{A}$*

Доведення теореми представлено у роботі [12].

3.2 Аналіз швикодії схем $AEAD$ на основі чорної-коробки

В додатках 3.3 приведено лістинг коду програми, за допомогою якої оцінюємо швидкодію різних комбінацій схем, а саме $CB(EAX, GCM)[AES], CB(CCM, GCM)[AES], CB(CCM, EAX)[AES]$ для довжини ключів 128 і 256 бітів, з nonce 96 бітів і тагом довжиною 128 бітів. Наведені заміри показують залежність між довжиною повідомлення(в байтах) та часом виконання алгоритму.

Технічні характеристики ЕОМ

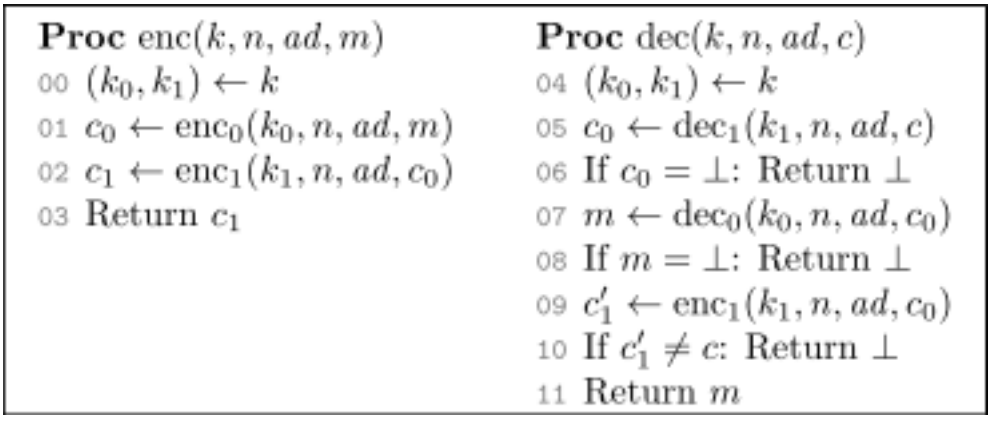


Рисунок 3.1 – Схема $AEAD$ на основі суматора "чорна-коробка".

$$\mathbf{CB}(AE_0, AE_1) = (enc, dec) : AE_i = (enc_i, dec_i), i \in \{0, 1\}$$

– 64 бітна архітектура

– Intel Core i5-7300HQ CPU @ 2.50Ghz × 4

На наведених графіках 3.2, 3.3 указані заміри для ключів 128 бітів і 256 відповідно. Як бачимо з графіків, є зони, в яких швидкодія виконання суттєво підстрибує чи навпаки падає. Це дія оптимізацій компілятора та технології JIT(just-in-time) оптимізації, тобто пришвидшення за рахунок кешування деяких даних процесором.

За допомогою спеціальних параметрів '-Xint' ми можемо відключати JIT оптимізацію. На графіках 3.4, 3.5 вказані відповідні заміри.

На даних графіках видно, що найбільш швидкою схемою для цієї архітектури є комбінація $CB(EAX, GCM)$. Зокрема з графіків можна зробити висновок, що схеми з використанням GCM швидші для 64-біт архітектур.

3.3 Підхід до побудови $AEAD$ суматорів з внутрішнім станом

В деяких випадках при використанні суматорів нам немає потреби масштабувати комбінацію, тобто збільшення кількості криптографічних примітивів, оскільки це збільшує кількість часу, витрачених на шифрування. Однак ми бажаємо, аби ідея використання суматорів зберіглась і комбінація вела себе так, ніби ми масштабували суматор, але без суттєвих затримок по часу внаслідок цього.

Нехай маємо функцію з внутрішнім станом $\rho : \{0,1\}^m \to \{0,1\}^{\|s\|_2}$, це може бути і лінійний регістр зсуву, який ми застосовуємо $\|s\|_2$ разів, де $\|\cdot\|_2$ довжина бітового рядка числа s в двійковій системі числення. І Маємо декілька схем $AEAD$ на основі суматорів $(CB_0, CB_1, ..CB_{s-1})$. Наприклад, $CB_0 = CB(GCM, EAX)$, $CB_1 = CB(GCM, CCM)$, $CB_2 = CB(CCM, EAX)$. Тоді сконструюємо схему :

Init

– $(k_1, k_2) \leftarrow k$

– Ініціалізуємо функцію $\rho(k_2)$

Шифрування $enc(k, n, ad, m)$

– $n \leftarrow \rho$

– $n := \|n\|_{10}$

– $c \leftarrow CB_n(k_1, n, ad, m))$

– Повернути значення c

Нехай задача супротивника в такому разі - вгадати n. Якщо функція ρ реалізує випадкову функцію, то ймовірність вгадати у супротивника $\frac{1}{n}$. Однак така схема має суттєвий недолік. Якщо на один з суматорів побудується успішна атака, то з результатів вимірювання швидкодії, вказаних в цій же роботі, і використанням атак по часу супротивник може кожного разу здогадуватись, яку з комбінацій було використано і таким чином вся схема може бути небезпечною.

Висновки до розділу 3

Отже, наведені результати показують, що схеми $AEAD$ на основі суматорів дають великий спектр робіт для аналізу швидкодії різних комбінацій примітивів. Підхід "чорної коробки"дає нам можилвість не втручаючись у схему автентифікованого шифрування отримати бажаний результат. Зокрема було надано приклад застосування результатів дослідження до атак по часу.

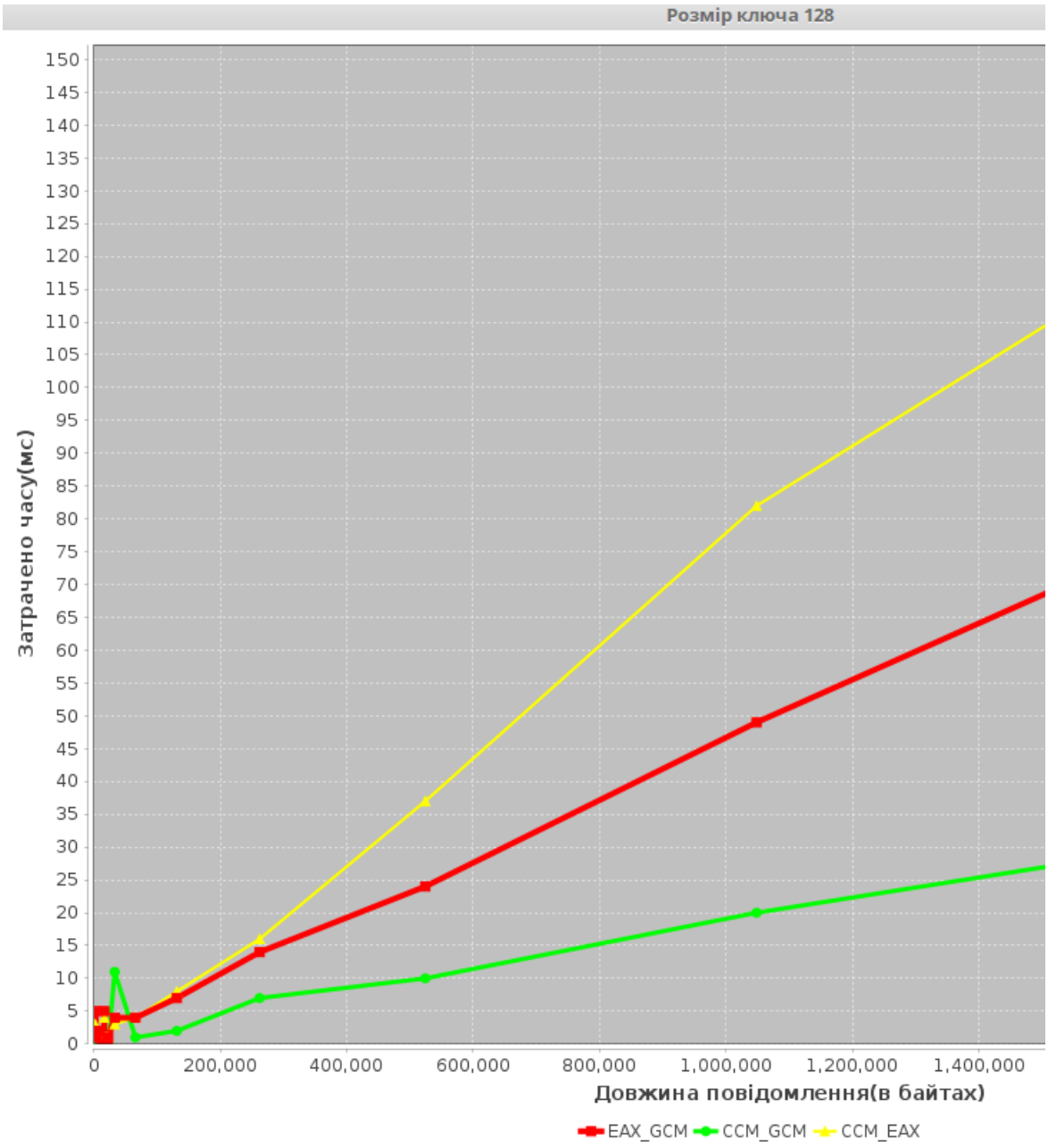


Рисунок 3.2 – Результат вимірювання для 128 бітного ключа

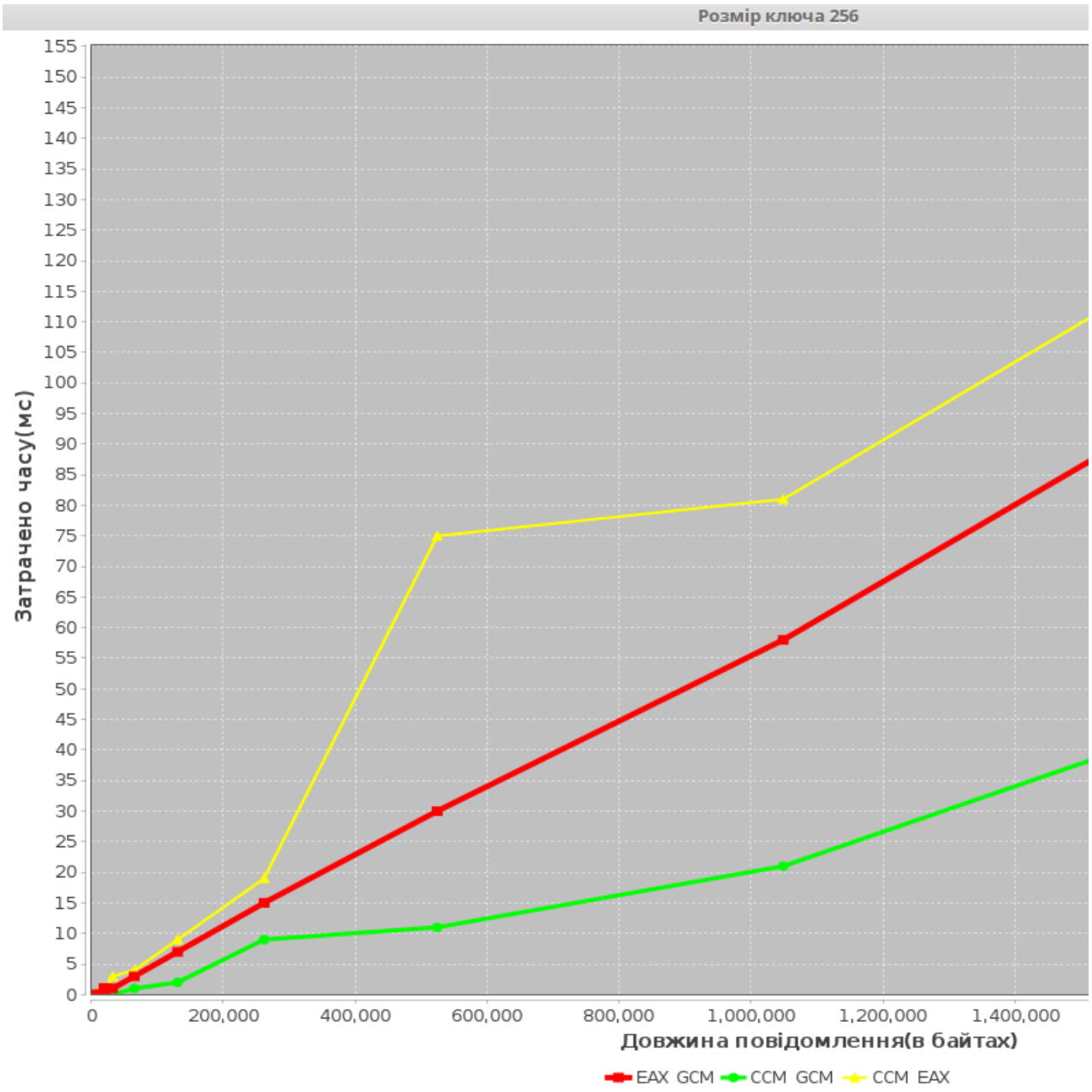


Рисунок 3.3 – Результат вимірювання для 256 бітного ключа

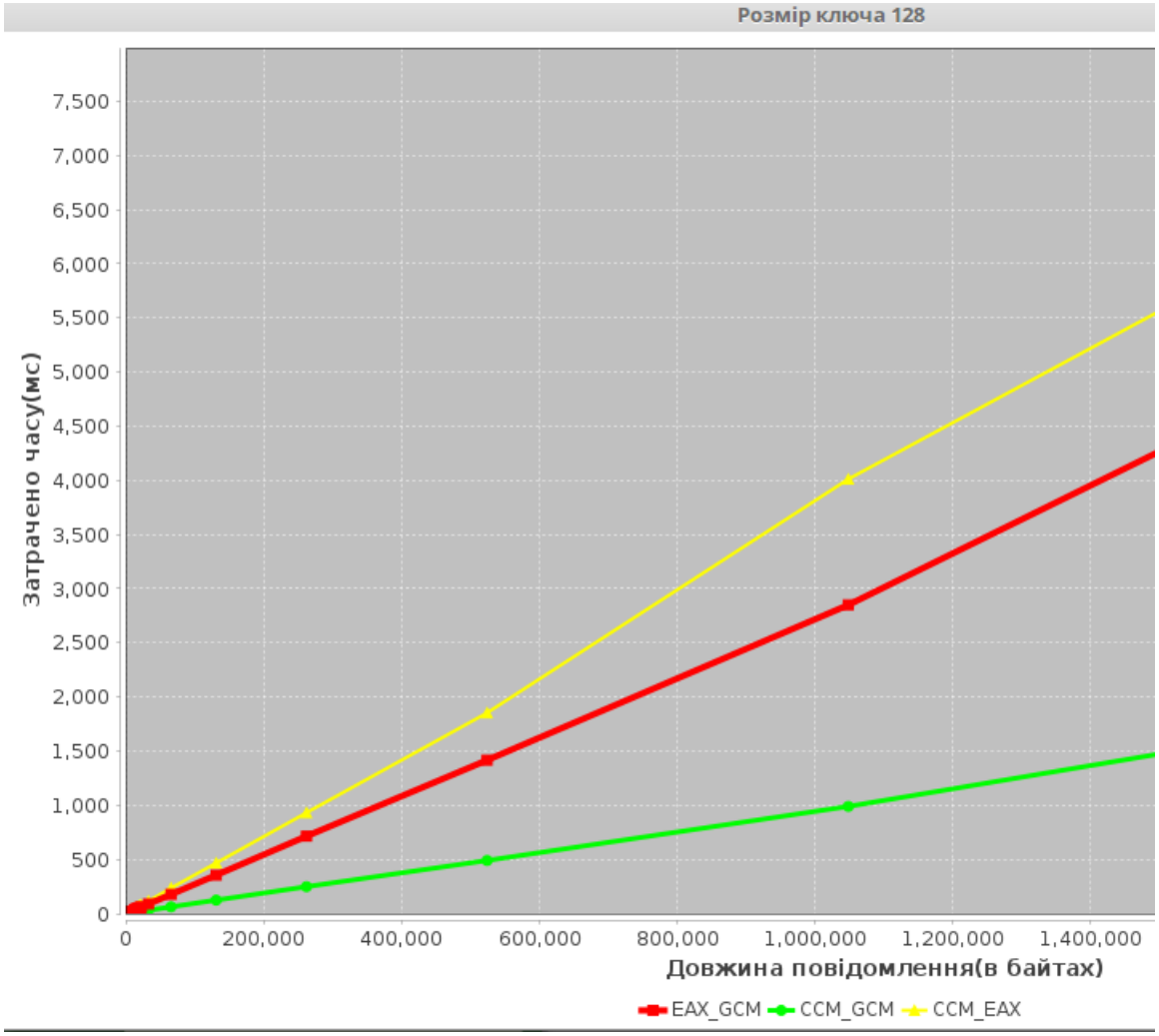


Рисунок 3.4 – Результат вимірювання для 128 бітного ключа без оптимізацій

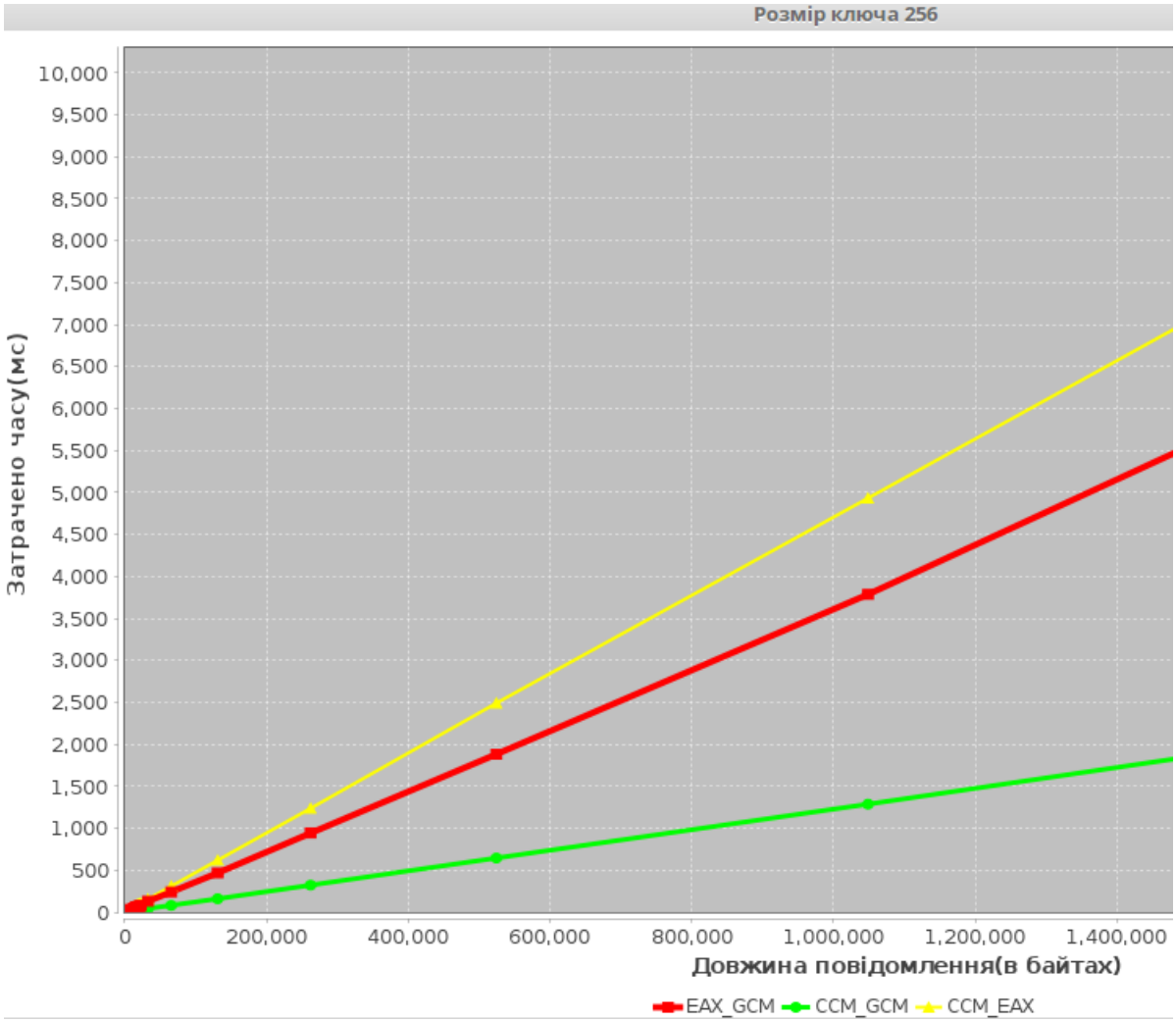


Рисунок 3.5 – Результат вимірювання для 256 бітного ключа без оптимізацій

ВИСНОВКИ

У ході даної роботи був проведений аналіз опублікованих джерел за такою тематикою, як підходи до побудови алгоритмів автентифікованого шифрування, узагальнена схема побудови та підхід з використанням суматорів, тобто комбінації існуючих схем $AEAD$. Зокрема були приведені аргументи щодо використання узагальненої схеми побудови $AEAD$ це $Encrypt - then - Mac(EtM)$, шла мова про більш сильні властивості, наприклад $CUF - CPA$, стійкість до підробки шифротексту, тобто якщо $Mac - then - Encrypt$ схема має таку властивість, то має гарну криптографічну стійкість і може використовуватись для побудови захищених каналів зв'язку. Зокрема було надано детальний огляд стійкості і швидкодії таких режимів $AEAD$, як GCM, EAX. Приведено підхід до побудови схеми $AEAD$ з суматорами і теорема про стійкість останніх. Зокрема була побудована програма лістинг коду якої вказаний в додатках, що використовує комбінацію з двох режимів автентифікованого шифрування, тобто скаладається з двох компонент. Під час побудови програми було використано середовище віртуальної машини $Java$ і внутрішні криптографічні бібліотеки мови, що дозволило зосередитись саме на підході 'чорної коробки', використовуючи уже наявні алгоритми. Подальші дослідження будуть акцентувати увагу на аналізі стійкості схем на основі суматорів з інтеграцією конкретних режимів.

ЛІТЕРАТУРА

[1] Mihir Bellare, Ran Canetti та Hugo Krawczyk. “Keying Hash Functions for Message Authentication”. в: CRYPTO ’96 (1996), с. 1—15.

[2] Mihir Bellare, Joe Kilian та Phillip Rogaway. “The Security of Cipher Block Chaining”. в: (серп. 1994), с. 341—358. DOI: 10.1007/3-540-48658-5_32.

[3] Mihir Bellare та Chanathip Namprempre. “Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm”. в: *J. Cryptol.* 21.4 (вер. 2008), с. 469—491. ISSN: 0933-2790. DOI: 10.1007/s00145-008-9026-x. URL: https://doi.org/10.1007/s00145-008-9026-x.

[4] Mihir Bellare, Phillip Rogaway та David Wagner. “The EAX Mode of Operation”. в: (2004). за ред. Bimal Roy та Willi Meier, с. 389—407.

[5] Daniel Bernstein. “Stronger Security Bounds for Wegman-Carter-Shoup Authenticators”. в: *Lecture Notes in Computer Science* 3494 (трав. 2005), с. 164—180. DOI: 10.1007/11426639_10.

[6] Ran Canetti та Hugo Krawczyk. “Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels”. в: *IACR Cryptol. ePrint Arch.* 2001 (2001), с. 40.

[7] Virgil Gligor та Pompiliu Donescu. “Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes”. в: *Lecture Notes in Computer Science* 1 (трав. 2001). DOI: 10.1007/3-540-45473-X_8.

[8] Akiko Inoue та ін. “Cryptanalysis of OCB2: Attacks on Authenticity and Confidentiality”. в: (серп. 2019), с. 3—31. DOI: 10.1007/978-3-030-26948-7_1.

[9] CS Jutla. “Encryption modes with almost free message integrity”. в: 2045 (січ. 2001), с. 529—544.

[10] Hugo Krawczyk. “The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?)” в: 2139 (лип. 2001). DOI: 10.1007/3-540-44647-8_19.

[11] Nicolas Meloni, Christophe Nègre та M. Hasan. “High Performance GHASH Function for Long Messages”. в: 6123 (черв. 2010), с. 154—167. DOI: 10.1007/978-3-642-13708-2_10.

[12] Bertram Poettering та Paul Rösler. “Combiners for AEAD”. в: *IACR Cryptol. ePrint Arch.* 2020 (2020), с. 232.

[13] Bart Prenel та Paul C. van Oorschot. “MDx-MAC and Building Fast MACs from Hash Functions”. в: CRYPTO ’95 (1995), с. 1—14.

[14] Phillip Rogaway. “Authenticated-encryption with associated-data”. в: (січ. 2002), с. 98—107. DOI: 10.1145/586110.586125.

[15] Phillip Rogaway та ін. “OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption”. в: *ACM Transactions on Information and System Security* 6 (січ. 2001), с. 196—205. DOI: 10.1145/501983.502011.

[16] Markku-Juhani Saarinen. “Cycling attacks on GCM, GHASH and other polynomial MACs and hashes”. в: (бер. 2012), с. 216—225. DOI: 10.1007/978-3-642-34047-5_13.

[17] M. N. Wegman та J. L. Carter. “New classes and applications of hash functions”. в: (1979), с. 175—182.

[18] Mark Wegman та J.Lawrence Carter. “New hash functions and their use in authentication and set equality”. в: *Journal of Computer and System Sciences* 22 (черв. 1981), с. 265—279. DOI: 10.1016/0022-0000(81)90033-7.

ДОДАТОК А ТЕКСТИ ПРОГРАМ

Тексти інструментальних програм для проведення експериментальних досліджень необхідно виносити у додатки.

А.1 Програма 1

```
class BlackBox extends AEAD {
    private AEAD A0;
    private AEAD A1;
    private byte[] mac;

    public BlackBox(AEAD A0, AEAD A1) {
        this.A0 = A0;
        this.A1 = A1;
    }

    @Override
    public byte[] encrypt(byte[] data) throws Exception {
        byte[] c0 =  encryptConcat(data, A0);
        byte[] c1 = encryptConcat(c0, A1);
        return c1;
    }

    @Override
    public byte[] getMac() {
        return mac;
    }

    private byte[] encryptConcat(byte[] data, AEAD A) throws Exception {
        byte[] encryptedData = A.encrypt(data);
        return concat(encryptedData, A.getMac());
    }

    private byte[] concat(byte[] a0, byte[] a1) {
        byte[] res = new byte[a0.length + a1.length];
        System.arraycopy(a0, 0, res, 0, a0.length);
        System.arraycopy(a1, 0, res, a0.length, a1.length);
        return res;
    }
}
```

```
public class CcmAes extends AEAD {
    public CcmAes(int macSize, byte[] nonce, byte[] key) {
        cipher = new CCMBlockCipher(new AESEngine());
        params = new CCMParameters(new KeyParameter(key), macSize, nonce, null);
        cipher.init(true, params);
    }
}

public class EaxAes extends AEAD {
    public EaxAes(int macSize, byte[] nonce, byte[] key) {
        cipher = new EAXBlockCipher(new AESEngine());
        params = new AEADParameters(new KeyParameter(key), macSize, nonce, null);
        cipher.init(true, params);
    }
}

public class GcmAes extends AEAD{
    public GcmAes(int macSize, byte[] nonce, byte[] key) {
        cipher = new GCMBlockCipher(new AESEngine());
        params = new AEADParameters(new KeyParameter(key), macSize, nonce, null);
        cipher.init(true, params);
    }
}

public abstract class AEAD {
    protected byte[] mac;
    protected AEADBlockCipher cipher;
    protected AEADParameters params;

    public byte[] encrypt(byte[] data) throws InvalidCipherTextException, Exception {
        byte[] outputText = new byte[cipher.getOutputSize(data.length)];
        int outputLen = cipher.processBytes(data, 0, data.length,
                outputText , 0);
        //cipher.doFinal(outputText, outputLen);
        this.mac = cipher.getMac();
        return outputText;
    }

    public byte[] getMac() {
        return mac;
    }
}
```